React 面试题

基本知识

1. 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。

2. 什么是 React？

React 是 React 是 facebook 在 2011 搞出来的一个轻量级的组件库，用于解决前端视图层的一些问题，就是 MVC 中 V 层的问题
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。

3、你了解 React 吗？/什么是 react?

答：了解，，它内部的 Instagram 网站就是用 React 搭建的。

4. React 有什么特点？

React 的主要功能如下：
它使用虚拟 DOM 而不是真正的 DOM。
它可以进行服务器端渲染。
它遵循单向数据流或数据绑定。

5、React 解决了什么问题？

答：解决了三个问题： 1.组件复用问题， 2.性能问题，3.兼容性问题：

6、React 的协议？

答：React 遵循的协议是“BSD 许可证 + 专利开源协议”，这个协议比较奇葩，如果你的产品跟 facebook 没有竞争关系，你可以自由的使用 react，但是如果有竞争关系，你的 react 的使用许可将会被取消

7、react 生命周期函数

这个问题要考察的是组件的生命周期

一、初始化阶段：

① getDefaultProps:获取实例的默认属性

② getInitialState:获取每个实例的初始化状态

③ componentWillMount：组件即将被装载、渲染到页面上

④ render:组件在这里生成虚拟的 DOM 节点

⑤ componentDidMount:组件真正在被装载之后[AJAX 请求]

二、运行中状态：

1．componentWillReceiveProps:组件将要接收到属性的时候调用

2．shouldComponentUpdate:组件接受到新属性或者新状态的时候（可以返回 false，接收数据后不更新，阻止 render 调用，后面的函数不会被继续执行了）

3．componentWillUpdate:组件即将更新不能修改属性和状态

4．render:组件重新描绘

5．componentDidUpdate:组件已经更新

三、销毁阶段：

componentWillUnmount:组件即将销毁

8、react 性能优化是哪个周期函数？

shouldComponentUpdate 这个方法用来判断是否需要调用 render 方法重新描绘 dom。因为 dom 的描绘非常消耗性能，如果我们能在 shouldComponentUpdate 方法中能够写出更优化的 dom diff 算法，可以极大的提高性能。

9、为什么虚拟 dom 会提高性能?

虚拟 dom 相当于在 js 和真实 dom 中间加了一个缓存，利用 dom diff 算法避免了没有必要的 dom 操作，从而提高性能。

具体实现步骤如下：
用 JavaScript 对象结构表示 DOM 树的结构；然后用这个树构建一个真正的 DOM 树，插到文档当中
当状态变更的时候，重新构造一棵新的对象树。然后用新的树和旧的树进行比较，记录两棵树差异
把 2 所记录的差异应用到步骤 1 所构建的真正的 DOM 树上，视图就更新了。

10. 列出 React 的一些主要优点。

React 的一些主要优点是：
1.只需查看 render 函数就会很容易知道一个组件是如何被渲染的
2.JSX 的引入，使得组件的代码更加可读，也更容易看懂组件的布局，或者组件之间是如何互相引用的
3.支持服务端渲染，这可以改进 SEO 和性能
4.易于测试
5.React 只关注 View 层，所以可以和其它任何框架(如 Angular.js)一起使用

11. React 有哪些限制？

React 的限制如下：
React 只是一个库，而不是一个完整的框架
它的库非常庞大，需要时间来理解
新手程序员可能很难理解
编码变得复杂，因为它使用内联模板和 JSX

12. 什么是 JSX?

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。下面是 JSX 的一个例子：

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

13、diff 算法?

把树形结构按照层级分解，只比较同级元素。
给列表结构的每个单元添加唯一的 key 属性，方便比较。
React 只会匹配相同 class 的 component（这里面的 class 指的是组件的名字）
合并操作，调用 component 的 setState 方法的时候, React 将其标记为 dirty.到每一个事件循环结束, React 检查所有标记 dirty 的 component 重新绘制.
选择性子树渲染。开发人员可以重写 shouldComponentUpdate 提高 diff 的性能。

14、了解 shouldComponentUpdate 吗？

答：React 虚拟 dom 技术要求不断的将 dom 和虚拟 dom 进行 diff 比较，如果 dom 树比价大，这种比较操作会比较耗时，因此 React 提供了 shouldComponentUpdate 这种补丁函数，如果对于一些变化，如果我们不希望某个组件刷新，或者刷新后跟原来其实一样，就可以使用这个函数直接告诉 React，省去 diff 操作，进一步的提高了效率。

15、React 的工作原理？

答：React 会创建一个虚拟 DOM(virtual DOM)。当一个组件中的状态改变时，React 首先会通过 "diffing" 算法来标记虚拟 DOM 中的改变，第二步是调节(reconciliation)，会用 diff 的结果来更新 DOM。

16、简述 flux 思想

Flux 的最大特点，就是数据的”单向流动”。
1.用户访问 View
2.View 发出用户的 Action
3.Dispatcher 收到 Action，要求 Store 进行相应的更新
4.Store 更新后，发出一个”change”事件
5.View 收到”change”事件后，更新页面

17、react 性能优化方案

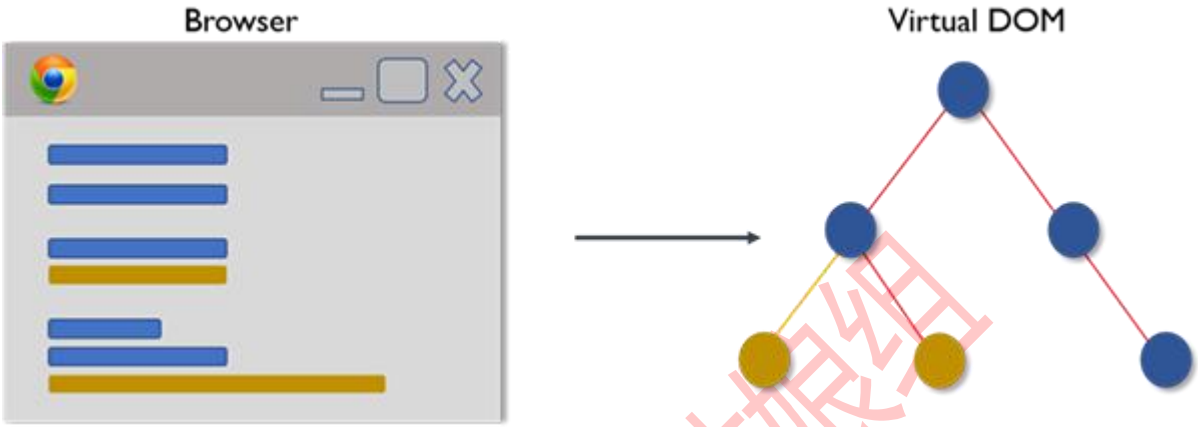
（1）重写 shouldComponentUpdate 来避免不必要的 dom 操作。
（2）使用 production 版本的 react.js。
（3）使用 key 来帮助 React 识别列表中所有子组件的最小变化。

18. 你了解 Virtual DOM 吗？解释一下它的工作原理。

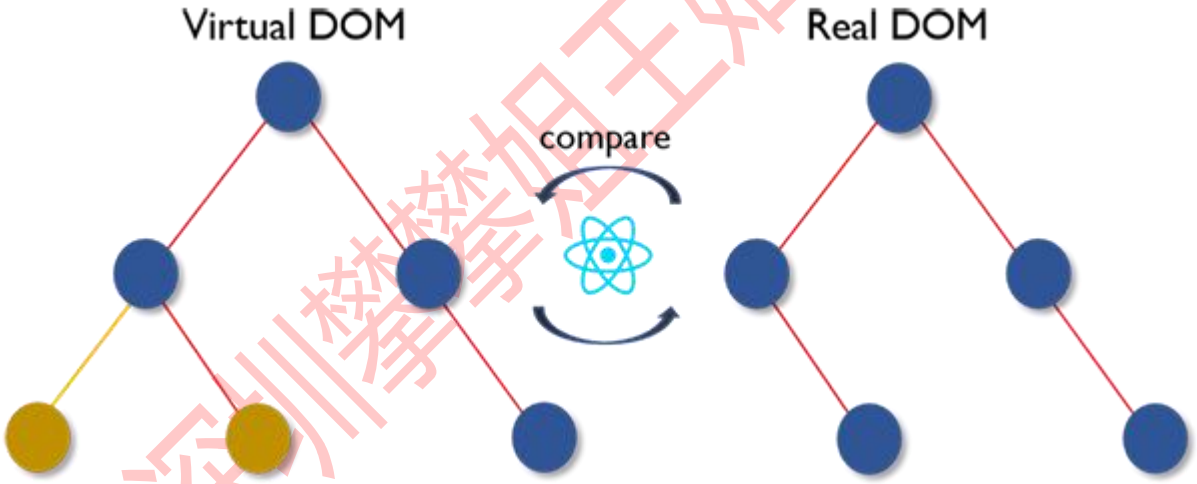
Virtual DOM 是一个轻量级的 JavaScript 对象，它最初只是 real DOM 的副本。它是一个节点树，它将元素、它们的属性和内容作为对象及其属性。 React 的渲染函数从 React 组件中创建一个节点树。然后它响应数据模型中的变化来更新该树，该变化是由用户或系统完成的各种动作引起的。

Virtual DOM 工作过程有三个简单的步骤。

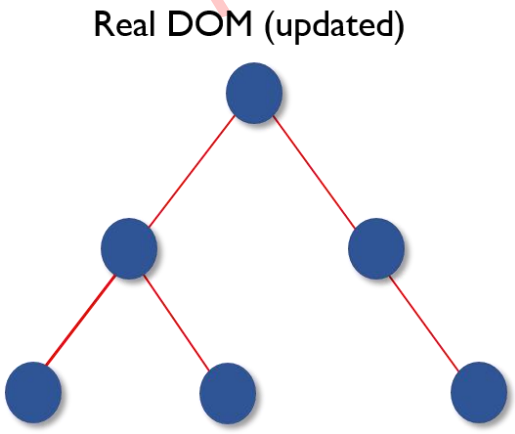
1.每当底层数据发生改变时，整个 UI 都将在 Virtual DOM 描述中重新渲染。



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更新 real DOM。



19. 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。

20. 与 ES5 相比，React 的 ES6 语法有何不同？

以下语法是 ES5 与 ES6 中的区别：

1.require 与 import

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

2.export 与 exports

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

3.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>;
  }
}
```

4.props

```
// ES5
var App = React.createClass({
  propTypes: { name: React.PropTypes.string },
```

```
  render: function() {
    return
      <h3>Hello, {this.props.name}!</h3>;
  }
});

// ES6
class App extends React.Component {
  render() {
    return
      <h3>Hello, {this.props.name}!</h3>;
  }
}
```

5.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>;
  }
}
```

21. 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

主题	React	Angular
2. 渲染	可以进行服务器端渲染	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

22. 使用箭头函数(arrow functions)的优点是什么？

答：1. 作用域安全：在箭头函数之前，每一个新创建的函数都有定义自身的 this 值(在构造函数中是新对象；在严格模式下，函数调用中的 this 是未定义的；如果函数被称为“对象方法”，则为基础对象等)，但箭头函数不会，它会使用封闭执行上下文的 this 值。
2. 简单：箭头函数易于阅读和书写
3. 清晰：当一切都是一个箭头函数，任何常规函数都可以立即用于定义作用域。开发者总是可以查找 next-higher 函数语句，以查看 this 的值

23、为什么建议传递给 setState 的参数是一个 callback 而不是一个对象？

答：因为 this.props 和 this.state 的更新可能是异步的，不能依赖它们的值去计算下一个 state。

24、除了在构造函数中绑定 this，还有其它方式吗？

答：可以使用属性初始值设定项(property initializers)来正确绑定回调，create-react-app 也是默认支持的。在回调中你可以使用箭头函数，但问题是每次组件渲染时都会创建一个新的回调。

25、怎么阻止组件的渲染？

答：在组件的 render 方法中返回 null 并不会影响触发组件的生命周期方法

26、当渲染一个列表时，何为 key？设置 key 的目的是什么？

答：Keys 会有助于 React 识别哪些 items 改变了，被添加了或者被移除了。Keys 应该被赋予数组内的元素以赋予(DOM)元素一个稳定的标识，选择一个 key 的最佳方法是使用一个字符串，该字符串能惟一地标识一个列表项。很多时候你会使用数据中的 IDs 作为 keys，

当你没有稳定的 IDs 用于被渲染的 items 时，可以使用项目索引作为渲染项的 key，但这种方式并不推荐，如果 items 可以重新排序，就会导致 re-render 变慢

27、(在构造函数中)调用 super(props) 的目的是什么？

答:在 super() 被调用之前,子类是不能使用 this 的,在 ES2015 中,子类必须在 constructor 中调用 super()。传递 props 给 super() 的原因则是便于(在子类中)能在 constructor 访问 this.props。

28.何为 Children ？

答：在 JSX 表达式中，一个开始标签(比如<a>)和一个关闭标签(比如)之间的内容会作为一个特殊的属性 props.children 被自动传递给包含着它的组件。
这个属性有许多可用的方法，包括 React.Children.map，React.Children.forEach，React.Children.count，React.Children.only，React.Children.toArray。

29、在 React 中，何为 state?

答：State 和 props 类似，但它是私有的，并且完全由组件自身控制。State 本质上是一个持有数据，并决定组件如何渲染的对象。

30、什么原因会促使你脱离 create-react-app 的依赖？

答；当你想去配置 webpack 或 babel presets。

31、何为 action ？

答；Actions 是一个纯 javascript 对象，它们必须有一个 type 属性表明正在执行的 action 的类型。实质上，action 是将数据从应用程序发送到 store 的有效载荷。

32、何为 reducer ？

答：一个 reducer 是一个纯函数，该函数以先前的 state 和一个 action 作为参数，并返回下一个 state。

33、何为纯函数(pure function)？

答：一个纯函数是一个不依赖于且不改变其作用域之外的变量状态的函数，这也意味着一个纯函数对于同样的参数总是返回同样的结果。

React 组件

34. 你怎样理解“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。

35. 怎样解释 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 <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。

36. 如何将两个或多个组件嵌入到一个组件中？

可以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')
);
```

37. 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。

38. 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。

39. 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

40. 如何更新组件的状态？

可以用 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')
);
```

41、(组件的)状态(state)和属性(props)之间有何不同？

答：1. State 是一种数据结构，用于组件挂载时所需数据的默认值。State 可能会随着时间的推移而发生突变，但多数时候是作为用户事件行为的结果。
2. Props(properties 的简写)则是组件的配置。props 由父组件传递给子组件，并且就子组件而言，props 是不可变的(immutable)。组件不能改变自身的 props，但是可以把其子组件的 props 放在一起(统一管理)。Props 也不仅仅是数据--回调函数也可以通过 props 传递。

42. 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) } />
  );
}
```

43. 区分有状态和无状态组件。

有状态组件	无状态组件
1. 在内存中存储有关组件状态变化的信息	1. 计算组件的内部的状态
2. 有权改变状态	2. 无权改变状态
3. 包含过去、现在和未来可能的状态变化情况	3. 不包含过去，现在和未来可能发生的状态变化情况
4. 接受无状态组件状态变化要求的通知，然后将 props 发送给他们。	4.从有状态组件接收 props 并将其视为回调函数。

43. 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***阶段，组件被销毁并从 DOM 中删除。

44. 详细解释 React 组件的生命周期方法。

一些最重要的生命周期方法是：
componentWillMount() - 在渲染之前执行，在客户端和服务器端都会执行。
componentDidMount() - 仅在***次渲染后在客户端执行。
componentWillReceiveProps() - 当从父类接收到 props 并且在调用另一个渲染器之前调用。
shouldComponentUpdate() - 根据特定条件返回 true 或 false。如果你希望更新组件，请返回 true 否则返回 false。默认情况下，它返回 false。
componentWillUpdate() - 在 DOM 中进行渲染之前调用。
componentDidUpdate() - 在渲染发生后立即调用。
componentWillUnmount() - 从 DOM 卸载组件后调用。用于清理内存空间。

45、在生命周期中的哪一步你应该发起 AJAX 请求？

我们应当将 AJAX 请求放到 componentDidMount 函数中执行，主要原因有下：
React 下一代调和算法 Fiber 会通过开始或停止渲染的方式优化应用性能，其会影响到 componentWillMount 的触发次数。对于 componentWillMount 这个生命周期函数的调用次数会变得不确定，React 可能会多次频繁调用 componentWillMount。如果我们将 AJAX 请

求放到 componentWillMount 函数中，那么显而易见其会被触发多次，自然也就不是好的选择。
如果我们将 AJAX 请求放置在生命周期的其他函数中，我们并不能保证请求仅在组件挂载完毕后才会要求响应。如果我们的数据请求在组件挂载之前就完成，并且调用了 setState 函数将数据添加到组件状态中，对于未挂载的组件则会报错。而在 componentDidMount 函数中进行 AJAX 请求则能有效避免这个问题。

46. React 中的事件是什么？

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，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。

47. 如何在 React 中创建一个事件？

```
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>
    );
  }
});
```

48. 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。

49、React 中 refs 的作用是什么？

Refs 是 React 提供给我们的安全访问 DOM 元素或者某个组件实例的句柄。我们可以为元素添加 ref 属性然后在回调函数中接受该元素在 DOM 树中的句柄，该值会作为回调函数的第一个参数返回

50. 列出一些应该使用 Refs 的情况。

以下是应该使用 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

51. 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>
    );
  }
}
```


52. 如何在 React 中创建表单

React 表单类似于 HTML 表单。但是在 React 中，状态包含在组件的 state 属性中，并且只能通过 setState() 更新。因此元素不能直接更新它们的状态，它们的提交是由 JavaScript 函数处理的。此函数可以完全访问用户输入到表单的数据。

```
handleSubmit(event) {
  alert('A name was submitted: ' + this.state.value);
  event.preventDefault();
}
```

```
render() {
  return (
    <form onSubmit={this.handleSubmit}>
      <label>
        Name:
        <input type="text" value={this.state.value} onChange={this.handleSubmit} />
      </label>
      <input type="submit" value="Submit" />
    </form>
  );
}
```

53. 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

54. 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。

55. 你能用 HOC 做什么？

HOC 可用于许多任务，例如：
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

56. 什么是纯组件？

纯（Pure）组件是可以编写的最简单、最快的组件。它们可以替换任何只有 render() 的组件。这些组件增强了代码的简单性和应用的性能。

57、展示组件(Presentational component)和容器组件(Container component)之间有何不同？

答：1.展示组件关心组件看起来是什么。展示专门通过 props 接受数据和回调，并且几乎不会有自身的状态，但当展示组件拥有自身的状态时，通常也只关心 UI 状态而不是数据的状态。
2.容器组件则更关心组件是如何运作的。容器组件会为展示组件或者其它容器组件提供数据和行为(behavior)，它们会调用 Flux actions，并将其作为回调提供给展示组件。容器组件经常是有状态的，因为它们是(其它组件的)数据源

58、类组件(Class component)和函数式组件(Functional component)之间有何不同？

答：1.类组件不仅允许你使用更多额外的功能，如组件自身的状态和生命周期钩子，也能使组件直接访问 store 并维持状态
2.当组件仅是接收 props，并将组件自身渲染到页面时，该组件就是一个 '无状态组件(stateless component)'，可以使用一个纯函数来创建这样的组件。这种组件也被称为哑组件(dumb components)或展示组件

59、React 中 key 的作用是什么？

Key 是 React 用于追踪哪些列表中元素被修改、被添加或者被移除的辅助标识。在开发过程中，我们需要保证某个元素的 key 在其同级元素中具有唯一性。在 React Diff 算法中 React 会借助元素的 Key 值来判断该元素是新近创建的还是被移动而来的元素，从而减少不必要的元素重渲染。此外，React 还需要借助 Key 值来判断元素与本地状态的关联关系，因此我们绝不可忽视转换函数中 Key 的重要性。

60. 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。

61、 Reactjs component 中 prop 和 state 的区别

props 放初始化数据，一直不变的，state 就是放要变的。
需要理解的是，props 是一个父组件传递给子组件的数据流，这个数据流可以一直传递到子

孙组件。而 state 代表的是一个组件内部自身的状态（可以是父组件、子孙组件）
改变一个组件自身状态，从语义上来说，就是这个组件内部已经发生变化，有可能需要对此组件以及组件所包含的子孙组件进行重渲染。
两者的变化都有可能导致组件重渲染
state：如果 component 的某些状态需要被改变，并且会影响到 component 的 render，那么这些状态就应该用 state 表示。例如：一个购物车的 component，会根据用户在购物车中添加的产品和产品数量，显示不同的价格，那么“总价”这个状态，就应该用 state 表示。
props:如果 component 的某些状态由外部所决定，并且会影响到 component 的 render，那么这些状态就应该用 props 表示。例如：一个下拉菜单的 component，有哪些菜单项，是由这个 component 的使用者和使用场景决定的，那么“菜单项”这个状态，就应该用 props 表示，并且由外部传入。

62、调用 setState 之后发生了什么？

在代码中调用 setState 函数之后，React 会将传入的参数对象与组件当前的状态合并，然后触发所谓的调和过程（Reconciliation）。经过调和过程，React 会以相对高效的方式根据新的状态构建 React 元素树并且着手重新渲染整个 UI 界面。在 React 得到元素树之后，React 会自动计算出新的树与老树的节点差异，然后根据差异对界面进行最小化重渲染。在差异计算算法中，React 能够相对精确地知道哪些位置发生了改变以及应该如何改变，这就保证了按需更新，而不是全部重新渲染。

63、shouldComponentUpdate 的作用是啥以及为何它这么重要？

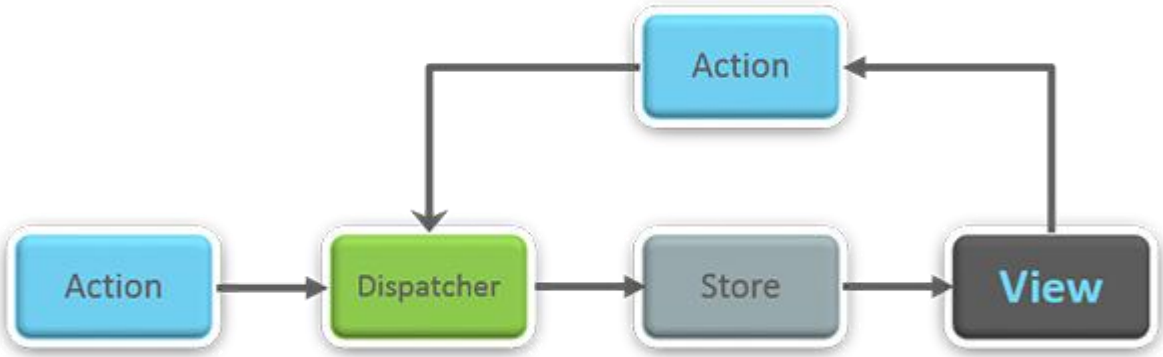
shouldComponentUpdate 允许我们手动地判断是否要进行组件更新，根据组件的应用场景设置函数的合理返回值能够帮我们避免不必要的更新。

React Redux

64. MVC 框架的主要问题是什么？

以下是 MVC 框架的一些主要问题：
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

65. 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。

66. 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。

67、在 Redux 中，何为 store ？

答：Store 是一个 javascript 对象，它保存了整个应用的 state。与此同时，Store 也承担以下职责：
允许通过 getState() 访问 state
运行通过 dispatch(action) 改变 state
通过 subscribe(listener) 注册 listeners
通过 subscribe(listener) 返回的函数处理 listeners 的注

68、Redux Thunk 的作用是什么？

答：Redux thunk 是一个允许你编写返回一个函数而不是一个 action 的 actions creators 的中间件。如果满足某个条件，thunk 则可以用来延迟 action 的派发(dispatch)，这可以处理异步 action 的派发(dispatch)。

69、redux 中间件

答：中间件提供第三方插件的模式，自定义拦截 action -> reducer 的过程。变为 action -> middlewares -> reducer 。这种机制可以让我们改变数据流，实现如异步 action ，action 过滤，日志输出，异常报告等功能。

常见的中间件： redux-logger：提供日志输出；redux-thunk：处理异步操作；redux-promise：处理异步操作；actionCreator 的返回值是 promise

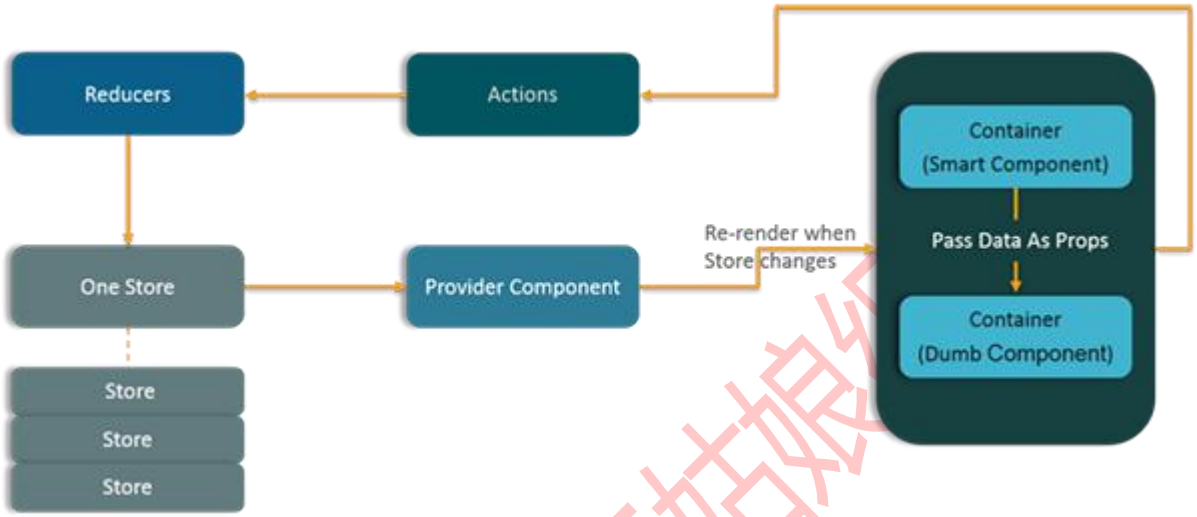
70. Redux 遵循的三个原则是什么？

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。

71. 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。
39. 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 – 只显示 Store 提供的数据。
40. 数据如何通过 Redux 流动？



72. 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 Creator 的示例：

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

73. 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。

74. 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。

75. 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

76. Redux 有哪些优点？

Redux 的优点如下：
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。

77、redux 有什么缺点

1.一个组件所需要的数据，必须由父组件传过来，而不能像 flux 中直接从 store 取。
2.当一个组件相关数据更新时，即使父组件不需要用到这个组件，父组件还是会重新 render，可能会有效率影响，或者需要写复杂的 shouldComponentUpdate 进行判断。

78、react 组件的划分业务组件技术组件？

根据组件的职责通常把组件分为 UI 组件和容器组件。

UI 组件负责 UI 的呈现，容器组件负责管理数据和逻辑。
两者通过 React-Redux 提供 connect 方法联系起来。
React 路由

79. 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。

80. 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***个匹配项后，它将渲染指定的路径。从而绕过其它路线。

81. 为什么需要 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 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，每个路由都会向我们提供一个独特的视图

```
<switch>
  <route exact path=’/’ component={Home}/>
  <route path=’/posts/:id’ component={Newpost}/>
  <route path=’/posts’ component={Post}/>
</switch>
```

82. 列出 React Router 的优点。

几个优点是：
就像 React 基于组件一样，在 React Router v4 中，API 是 'All About Components'。可以将 Router 可视化为单个根组件（<BrowserRouter>），其中我们将特定的子路由（<route>）包起来。
无需手动设置历史值：在 React Router v4 中，我们要做的就是将路由包装在 <BrowserRouter> 组件中。
包是分开的：共有三个包，分别用于 Web、Native 和 Core。这使我们应用更加紧凑。基于类似的编码风格很容易进行切换。

83. React Router 与常规路由有何不同？

主题	常规路由	React 路由
参与的页面	每个视图对应一个新文件	只涉及单个 HTML 页面
URL 更改	HTTP 请求被发送到服务器并且接收相应的 HTML 页面	仅更改历史记录属性
体验	用户实际在每个视图的不同页面切换	用户认为自己正在不同的页面间切换