匠人学院
jiangren.com.au

React JS
面试宝典

匠人学院™

jiangren.com.au

匠人学院成立于2017年，致力于帮助在澳洲华人，让找工作不再是难事，是澳洲学习和工作伙伴。超过10000的小伙伴加入我们，三年服务超过了5000名学生，帮助华人从学业困难，到就业困难，超过800多位同学拿到了Offer，进入了澳洲主流社会。我们希望成为下一代的肩膀，帮助更多的华人解决问题，也希望有同样目标，志同道合的人加入。

10000+ 成员　导师 300+

80 家本地企业　offer数量 800+

每年 100+ 场活动

Level 13b, 116 Adelaide Street, Brisbane CBD

Level 8, 11 York St, Wynyard, Sydney CBD

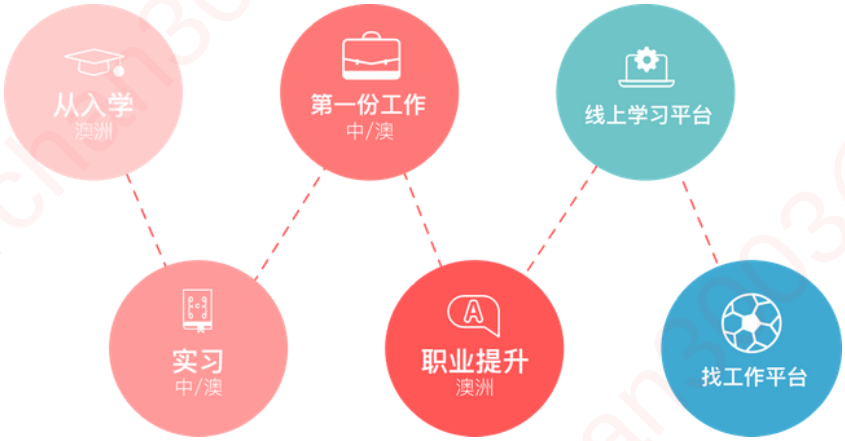
Suite 4.03, 838 Collins St, Docklands, Melbourne

Business Hub, 155 Waymouth St, Adelaide

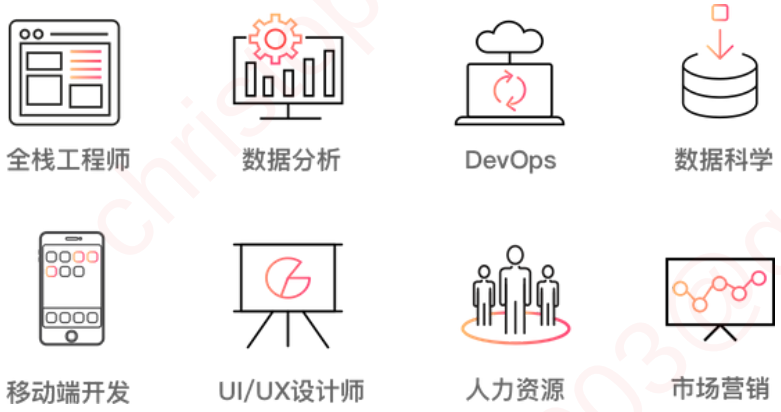
学IT找匠人

匠人学院™
jiangren.com.au

从学业，实习，到就业培训，公司猎头，以及学习平台，更多全方位帮助学生



覆盖软件开发、Web开发、UI/UX、运维、人力资源、移动端开发、数据分析、数据科学、市场营销。



学IT找匠人

面试题

Q1. How React works? How Virtual-DOM works in React?

React creates a virtual DOM. When state changes in a component it firstly runs a “diffing” algorithm, which identifies what has changed in the virtual DOM. The second step is reconciliation, where it updates the DOM with the results of diff.

The HTML DOM is always tree-structured — which is allowed by the structure of HTML document. The DOM trees are huge nowadays because of large apps. Since we are more and more pushed towards dynamic web apps (Single Page Applications — SPAs), we need to modify the DOM tree incessantly and a lot. And this is a real performance and development pain.

The Virtual DOM is an abstraction of the HTML DOM. It is lightweight and detached from the browser-specific implementation details. It is not invented by React but it uses it and provides it for free. ReactElements lives in the virtual DOM. They make the basic nodes here. Once we defined the elements, ReactElements can be render into the "real" DOM.

Whenever a ReactComponent is changing the state, diff algorithm in React runs and identifies what has changed. And then it updates the DOM with the results of diff. The point is - it’s done faster than it would be in the regular DOM.

Q2. What is JSX?

JSX is a syntax extension to JavaScript and comes with the full power of JavaScript. JSX produces React “elements”. You can embed any JavaScript expression in JSX by wrapping it in curly braces. After compilation, JSX expressions become regular JavaScript objects. This means that you can use JSX inside of if statements and for loops, assign it to variables, accept it as arguments, and return it from functions. Eventhough React does not require JSX, it is the recommended way of describing our UI in React app.

For example, below is the syntax for a basic element in React with JSX and its equivalent without it.

```
const element = (
  <h1 className="greeting">
    Hello, world!
  </h1>
);
```

Equivalent of the above using React.createElement

```
const element = React.createElement(
  'h1',
  {"className": "greeting"},
  'Hello, world!'
);
```

Q3. What is React.createClass?

React.createClass allows us to generate component "classes." But with ES6, React allows us to implement component classes that use ES6 JavaScript classes. The end result is the same -- we have a component class. But the style is different. And one is using a "custom" JavaScript class system (createClass) while the other is using a "native" JavaScript class system.

When using React’s createClass() method, we pass in an object as an argument. So we can write a component using createClass that looks like this:

```
import React from 'react';

const Contacts = React.createClass({
  render() {
    return (
      <div></div>
    );
  }
});

export default Contacts;
```

Using an ES6 class to write the same component is a little different. Instead of using a method from the react library, we extend an ES6 class that the library defines, Component.

```
import React from 'react';

class Contacts extends React.Component({
  constructor(props) {
    super(props);
  }
  render() {
    return (
      <div></div>
    );
  }
}

export default Contacts;
```

constructor() is a special function in a JavaScript class. JavaScript invokes constructor() whenever an object is created via a class.

面试题

Q4. What is ReactDOM and what is the difference between ReactDOM and React?

Prior to v0.14, all ReactDOM functionality was part of React. But later, React and ReactDOM were split into two different libraries.

As the name implies, ReactDOM is the glue between React and the DOM. Often, we will only use it for one single thing: mounting with ReactDOM. Another useful feature of ReactDOM is ReactDOM.findDOMNode() which we can use to gain direct access to a DOM element.

For everything else, there’ s React. We use React to define and create our elements, for lifecycle hooks, etc. i.e. the guts of a React application.

Q5. What are the differences between a class component and functional component?

Class components allows us to use additional features such as local state and lifecycle hooks. Also, to enable our component to have direct access to our store and thus holds state.

When our component just receives props and renders them to the page, this is a ‘stateless component’, for which a pure function can be used. These are also called dumb components or presentational components.

From the previous question, we can say that our Booklist component is functional components and are stateless.

```
// BookList.js

import React from "react";

const Booklist = books => (
  <ul>
    {books.map(({ title, author }) =>

•  {title}-{author}

    )}
  </ul>
)
```

On the other hand, the BookListContainer component is a class component.

Q6. What is the difference between state and props?

The state is a data structure that starts with a default value when a Component mounts. It may be mutated across time, mostly as a result of user events.

Props (short for properties) are a Component’ s configuration. Props are how components talk to each other. They are received from above component and immutable as far as the Component receiving them is concerned. A Component cannot change its props, but it is responsible for putting together the props of its child Components. Props do not have to just be data — callback functions may be passed in as props.

There is also the case that we can have default props so that props are set even if a parent component doesn’ t pass props down.

```
Class SearchBar extends Component {
  constructor(props) {
    super(props);
    this.state = { term: '' };
  }
  render() {
    return (
      <div className="search-bar">
      <input
        value={this.state.term}
        onChange={event => this.onInputChange(event.target.value)} />
      </div>
    );
  }
  onInputChange(term) {
    this.setState({term});
    this.props.onSearchTermChange(term);
  }
}
```

Props and State do similar things but are used in different ways. The majority of our components will probably be stateless. Props are used to pass data from parent to child or by the component itself. They are immutable and thus will not be changed. State is used for mutable data, or data that will change. This is particularly useful for user input.

Q7. What are controlled components?

In HTML, form elements such as <input>, <textarea>, and <select> typically maintain their own state and update it based on user input. When a user submits a form the values from the aforementioned elements are sent with the form. With React it works differently. The component containing the form will keep track of the value of the input in it's state and will re-render the component each time the callback function e.g. onChange is fired as the state will be updated. A form element whose value is controlled by React in this way is called a "controlled component".

With a controlled component, every state mutation will have an associated handler function. This makes it straightforward to modify or validate user input.

Q8. What is a higher order component?

A higher-order component (HOC) is an advanced technique in React for reusing component logic. HOCs are not part of the React API. They are a pattern that emerges from React’ s compositional nature.

A higher-order component is a function that takes a component and returns a new component.

HOC’ s allow you to reuse code, logic and bootstrap abstraction. HOCs are common in third-party React libraries. The most common is probably Redux’ s connect function. Beyond simply sharing utility libraries and simple composition, HOCs are the best way to share behavior between React Components. If you find yourself writing a lot of code in different places that does the same thing, you may be able to refactor that code into a reusable HOC.

Q9. What is create-react-app?

create-react-app is the official CLI (Command Line Interface) for React to create React apps with no build configuration.
We don’ t need to install or configure tools like Webpack or Babel. They are preconfigured and hidden so that we can focus on the code. We can install easily just like any other node modules. Then it is just one command to start the React project.

```
create-react-app my-app
```

It includes everything we need to build a React app:

- React, JSX, ES6, and Flow syntax support.
- Language extras beyond ES6 like the object spread operator.
- Autoprefixed CSS, so you don’ t need -webkit- or other prefixes.
- A fast interactive unit test runner with built-in support for coverage reporting.

· A live development server that warns about common mistakes.
· A build script to bundle JS, CSS, and images for production, with hashes and sourcemaps.

Q10. What is Redux?

The basic idea of Redux is that the entire application state is kept in a single store. The store is simply a javascript object. The only way to change the state is by firing actions from your application and then writing reducers for these actions that modify the state. The entire state transition is kept inside reducers and should not have any side-effects.

Redux is based on the idea that there should be only a single source of truth for your application state, be it UI state like which tab is active or Data state like the user profile details.

```
{
  first_name: 'John',
  last_name: 'Doe',
  age: 28
}
```

All of these data is retained by redux in a closure that redux calls a store . It also provides us a recipe of creating the said store, namely createStore(x).

The createStore function accepts another function, x as an argument. The passed in function is responsible for returning the state of the application at that point in time, which is then persisted in the store. This passed in function is known as the reducer.

This is a valid example reducer function:

```
export default function reducer(state={}, action) {
  return state;
}
```

This store can only be updated by dispatching an action. Our App dispatches an action, it is passed into reducer; the reducer returns a fresh instance of the state; the store notifies our App and it can begin it's re render as required.

Q11. What is Redux Thunk used for?

Redux thunk is middleware that allows us to write action creators that return a function instead of an action. The thunk can then be used to delay the dispatch of an action if a certain condition is met. This allows us to handle the asyncronous dispatching of actions. The inner function receives the store methods dispatch and getState as parameters.

To enable Redux Thunk, we need to use applyMiddleware() as below

```
import { createStore, applyMiddleware } from 'redux';
import thunk from 'redux-thunk';
import rootReducer from './reducers/index';

// Note: this API requires redux@>=3.1.0
const store = createStore(
  rootReducer,
  applyMiddleware(thunk)
);
```

Q12. What is PureComponent? When to use PureComponent over Component?

PureComponent is exactly the same as Component except that it handles the shouldComponentUpdate method for us. When props or state changes, PureComponent will do a shallow comparison on both props and state. Component on the other hand won't compare current props and state to next out of the box. Thus, the component will re-render by default whenever shouldComponentUpdate is called.

When comparing previous props and state to next, a shallow comparison will check that primitives have the same value (eg, 1 equals 1 or that true equals true) and that the references are the same between more complex javascript values like objects and arrays.

It is good to prefer PureComponent over Component whenever we never mutate our objects.

```
class MyComponent extends React.PureComponent {

  render() {
    return <div>Hello World!</div>
  }

}
```

Q13. How Virtual-DOM is more efficient than Dirty checking?

In React, each of our components have a state. This state is like an observable. Essentially, React knows when to re-render the scene because it is able to observe when this data changes. Dirty checking is slower than observables because we must poll the data at a regular interval and check all of the values in the data structure recursively. By comparison, setting a value on the state will signal to a listener that some state has changed, so React can simply listen for change events on the state and queue up re-rendering.

The virtual DOM is used for efficient re-rendering of the DOM. This isn’t really related to dirty checking your data. We could re-render using a virtual DOM with or without dirty checking. In fact, the diff algorithm is a dirty checker itself.

We aim to re-render the virtual tree only when the state changes. So using an observable to check if the state has changed is an efficient way to prevent unnecessary re-renders, which would cause lots of unnecessary tree diffs. If nothing has changed, we do nothing.

Q14. Is setState() is async? Why is setState() in React Async instead of Sync?

setState() actions are asynchronous and are batched for performance gains. This is explained in documentation as below.

setState() does not immediately mutate this.state but creates a pending state transition. Accessing this.state after calling this method can potentially return the existing value. There is no guarantee of synchronous operation of calls to setState and calls may be batched for performance gains.

This is because setState alters the state and causes rerendering. This can be an expensive operation and making it synchronous might leave the browser unresponsive. Thus the setState calls are asynchronous as well as batched for better UI experience and performance.

Q15. What is render() in React? And explain its purpose?

Each React component must have a render() mandatorily. It returns a single React element which is the representation of the native DOM component. If more than one HTML element needs to be rendered, then they must be grouped together inside one enclosing tag such as <form>, <group>, etc. This function must be kept pure i.e., it must return the same result each time it is invoked.

Q16. What are controlled and uncontrolled components in React?

This relates to stateful DOM components (form elements) and the difference:

· A Controlled Component is one that takes its current value through props and notifies changes through callbacks like onChange. A parent component “controls” it by handling the callback and managing its own state and passing the new values as props to the controlled component. You could also call this a “dumb component”.

· A Uncontrolled Component is one that stores its own state internally, and you query the DOM using a ref to find its current value when you need it. This is a bit more like traditional HTML.

In most (or all) cases we should use controlled components.

Q17. Explain the components of Redux.

Redux is composed of the following components:

· Action — Actions are payloads of information that send data from our application to our store. They are the only source of information for the store. We send them to the store using store.dispatch(). Primarly, they are just an object describes what happened in our app.

· Reducer — Reducers specify how the application’s state changes in response to actions sent to the store. Remember that actions only describe what happened, but don’t describe how the application’s state changes. So this place determines how state will change to an action.

· Store — The Store is the object that brings Action and Reducer together. The store has the following responsibilities: Holds application state; Allows access to state via getState(); Allows state to be updated via dispatch(action); Registers listeners via subscribe(listener); Handles unregistering of listeners via the function returned by subscribe(listener).

It’s important to note that we’ll only have a single store in a Redux application. When we want to split your data handling logic, we’ll use reducer composition instead of many stores.

Q18. What is React.cloneElement? And the difference with this.props.children?

React.cloneElement clone and return a new React element using using the passed element as the starting point. The resulting element will have the original element's props with the new props merged in shallowly. New children will replace existing children. key and ref from the original element will be preserved.

React.cloneElement only works if our child is a single React element. For almost everything {this.props.children} is the better solution. Cloning is useful in some more advanced scenarios, where a parent send in an element and the child component needs to change some props on that element or add things like ref for accessing the actual DOM element.

Q19. What is the second argument that can optionally be passed to setState and what is its purpose?

A callback function which will be invoked when setState has finished and the component is re-rendered.

Since the setState is asynchronous, which is why it takes in a second callback function. With this function, we can do what we want immediately after state has been updated.

Q20. What is the difference between React Native and React?

React is a JavaScript library, supporting both front end web and being run on the server, for building user interfaces and web applications.

On the other hand, React Native is a mobile framework that compiles to native app components, allowing us to build native mobile applications (iOS, Android, and Windows) in JavaScript that allows us to use ReactJS to build our components, and implements ReactJS under the hood.

With React Native it is possible to mimic the behavior of the native app in JavaScript and at the end, we will get platform specific code as the output. We may even mix the native code with the JavaScript if we need to optimize our application further.

面试题

Q 21. What are the major features of React?

The major features of React are:

· It uses VirtualDOM instead RealDOM considering that RealDOM manipulations are expensive.
· Supports server-side rendering.
· Follows Unidirectional* data flow or data binding.
· Uses reusable/composable UI components to develop the view.

Q22. What are the advantages of using React?

· It is easy to know how a component is rendered, you just need to look at the render function.
· JSX makes it easy to read the code of your components. It is also really easy to see the layout, or how components are plugged/combined with each other.
· You can render React on the server-side. This enables improves SEO and performance.
· It is easy to test.
· You can use React with any framework (Backbone.js, Angular.js) as it is only a view layer.

Q23. What is the difference between HTML and React event handling?

1. In HTML, the event name should be in *lowercase*:

```
<button onclick='activateLasers()'>
```

Whereas in React it follows *camelCase* convention:

```
<button onClick={activateLasers}>
```

1. In HTML, you can return `false` to prevent default behavior:

```
<a href='#' onclick='console.log("The link was clicked."); return false;' />
```

Whereas in React you must call `preventDefault()` explicitly:

```
function handleClick(event) {
  event.preventDefault()
  console.log('The link was clicked.')
}
```

Q24. How to create components in React?

There are two possible ways to create a component.

1. **Function Components:** This is the simplest way to create a component. Those are pure JavaScript functions that accept props object as first parameter and return React elements:

```
function Greeting({ message }) {
  return <h1>{`Hello, ${message}`}</h1>
}
```

2. **Class Components:** You can also use ES6 class to define a component. The above function component can be written as:

```
class Greeting extends React.Component {
  render() {
    return <h1>{`Hello, ${this.props.message}`}</h1>
  }
}
```

Q25. What is the difference between a Presentational component and a Container component?

Presentational components are concerned with how things look. They generally receive data and callbacks exclusively via props. These components rarely have their own state, but when they do it generally concerns UI state, as opposed to data state.
Container components are more concerned with how things work. These components provide the data and behavior to presentational or other container components. They call Flux actions and provide these as callbacks to the presentational components. They are also often stateful as they serve as data sources.

Q26. Why should we not update the state directly?

If you try to update state directly then it won't re-render the component.

```
//Wrong
this.state.message = 'Hello world'
```

Instead use `setState()` method. It schedules an update to a component's state object. When state changes, the component responds by re-rendering.

```
//Correct
this.setState({ message: 'Hello World' })
```

Note: You can directly assign to the state object either in constructor or using latest javascript's class field declaration syntax.

Q27. What is the difference of lifecycle methods in React?

React 16.3+

· **getDerivedStateFromProps**: Invoked right before calling render() and is invoked on every render. This exists for rare use cases where you need derived state. Worth reading if you need derived state.
· componentDidMount: Executed after first rendering and here all AJAX requests, DOM or state updates, and set up event listeners should occur.

· **shouldComponentUpdate**: Determines if the component will be updated or not. By default it returns true. If you are sure that the component doesn't need to render after state or props are updated, you can return false value. It is a great place to improve performance as it allows you to prevent a re-render if component receives new prop.
· **getSnapshotBeforeUpdate**: Executed right before rendered output is committed to the DOM. Any value returned by this will be passed intocomponentDidUpdate(). This is useful to capture information from the DOM i.e. scroll position.
· **componentDidUpdate**: Mostly it is used to update the DOM in response to prop or state changes. This will not fire if shouldComponentUpdate() returnsfalse.
· **componentWillUnmount** It will be used to cancel any outgoing network requests, or remove all event listeners associated with the component.

Before 16.3

· **componentWillMount**: Executed before rendering and is used for App level configuration in your root component.
· componentDidMount: Executed after first rendering and here all AJAX requests, DOM or state updates, and set up event listeners should occur.
· **componentWillReceiveProps**: Executed when particular prop updates to trigger state transitions.
· **shouldComponentUpdate**: Determines if the component will be updated or not. By default it returns true. If you are sure that the component doesn't need to render after state or props are updated, you can return false value. It is a great place to improve performance as it allows you to prevent a re-render if component receives new prop.
· **componentWillUpdate**: Executed before re-rendering the component when there are props & state changes confirmed by shouldComponentUpdate() which returns true.
· **componentDidUpdate**: Mostly it is used to update the DOM in response to prop or state changes.
· **componentWillUnmount**: It will be used to cancel any outgoing network requests, or remove all event listeners associated with the component.

面试题

Q28. Where in a React component should you make an AJAX request?

componentDidMount is where an AJAX request should be made in a React component. This method will be executed when the component “mounts” (is added to the DOM) for the first time. This method is only executed once during the component’s life. Importantly, you can’t guarantee the AJAX request will have resolved before the component mounts. If it doesn't, that would mean that you’d be trying to setState on an unmounted component, which would not work. Making your AJAX request incomponentDidMount will guarantee that there’s a component to update.

Q29. What are refs used for in React?

Refs are used to get reference to a DOM node or an instance of a component in React. Good examples of when to use refs are for managing focus/text selection, triggering imperative animations, or integrating with third-party DOM libraries. You should avoid using string refs and inline ref callbacks. Callback refs are advised by React.

Q30. What is the alternative of binding `this` in the constructor?

You can use property initializers to correctly bind callbacks. This is enabled by default in create react app. You can use an arrow function in the callback. The problem here is that a new callback is created each time the component renders.

Q31. What is the purpose of super(props)?

A child class constructor cannot make use of this until super() has been called. Also, ES2015 class constructors have to call super() if they are subclasses. The reason for passing props to super() is to enable you to access this.props in the constructor.

Q32. How would you prevent a component from rendering?

Returning null from a component's render method does not affect the firing of the component's lifecycle methods.

Q33. Why is it advised to pass a callback function to setState as opposed to an object?

Because this.props and this.state may be updated asynchronously, you should not rely on their values for calculating the next state.

Q34. What are stateless components?

If the behaviour is independent of its state then it can be a stateless component. You can use either a function or a class for creating stateless components. But unless you need to use a lifecycle hook in your components, you should go for function components. There are a lot of benefits if you decide to use function components here; they are easy to write, understand, and test, a little faster, and you can avoid the this keyword altogether.

Q35. What are stateful components?

If the behaviour of a component is dependent on the state of the component then it can be termed as stateful component. These stateful components are always class components and have a state that gets initialized in the constructor.

```
class App extends Component {
  constructor(props) {
    super(props)
    this.state = { count: 0 }
  }

  render() {
    // ...
  }
}
```

面试题

Q36. How would you prevent a component from rendering?

Returning null from a component's render method means nothing will be displayed, but it does not affect the firing of the component's lifecycle methods.

If the amount of times the component re-renders is an issue, there are two options available. Manually implementing a check in the shouldComponentUpdate lifecycle method hook.

```
shouldComponentUpdate(nextProps, nextState){
  // Do some check here
  return resultOFCheckAsBoolean
}
```

Or using React.PureComponent instead of React.Component React.PureComponent implements shouldComponentUpdate() with a shallow prop and state comparison. This enables you to avoid re-rendering the component with the same props and state.

Q37. When rendering a list what is a key and what is it's purpose?

Keys help React identify which items have changed, are added, or are removed. Keys should be given to the elements inside the array to give the elements a stable identity. The best way to pick a key is to use a string that uniquely identifies a list item among its siblings. Most often you would use IDs from your data as keys. When you don't have stable IDs for rendered items, you may use the item index as a key as a last resort. It is not recommend to use indexes for keys if the items can reorder, as that would be slow.

面试题

Q38. 区分Real DOM和Virtual DOM

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。

Q39. React有什么特点?

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

Q40. React有什么优点?

React的一些主要优点是：
1. 它提高了应用的性能
2. 可以方便地在客户端和服务器端使用
3. 由于 JSX，代码的可读性很好
4. React 很容易与 Meteor，Angular 等其他框架集成
5. 使用React，编写UI测试用例变得非常容易

Q41. React有哪些限制?

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

Q42. 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。

Q43. 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

```
1.  // ES5
2.  var React = require('react');
3.
4.  // ES6
5.  import React from 'react';
```

面试题

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>;
    }
}
```

Q44. 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

面试题

Q45. 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。

Q46. 怎样解释 React 中 render() 的目的。

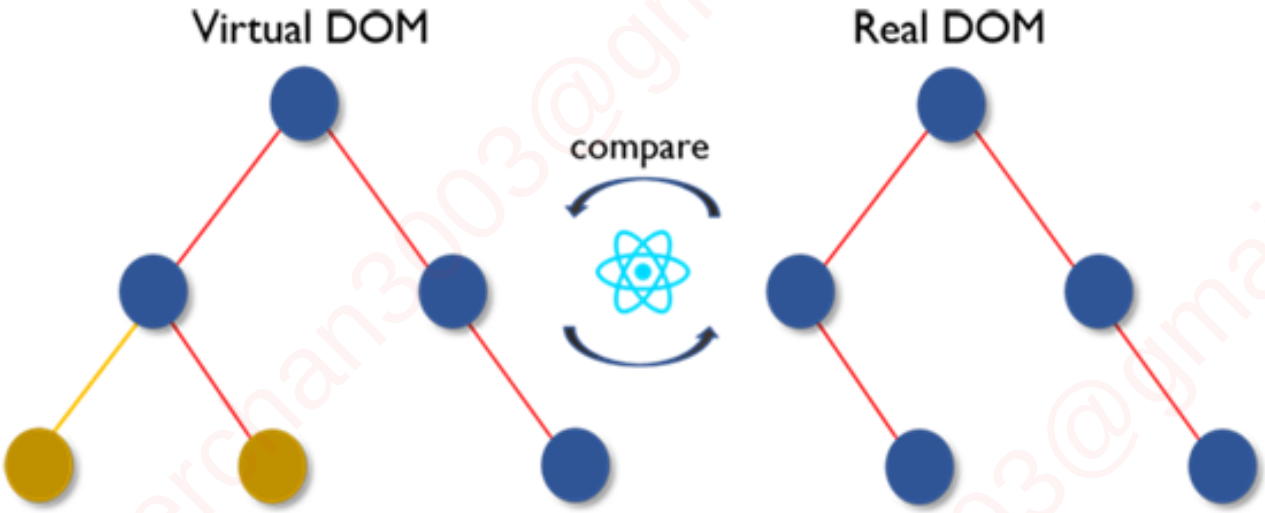
每个React组件强制要求必须有一个 render()。它返回一个 React 元素，是原生 DOM 组件的表示。如果需要渲染多个 HTML 元素，则必须将它们组合在一个封闭标记内，例如 <form>、<group>、 等。此函数必须保持纯净，即必须每次调用时都返回相同的结果。

Q47. 你了解 Virtual DOM 吗？解释一下它的工作原理。

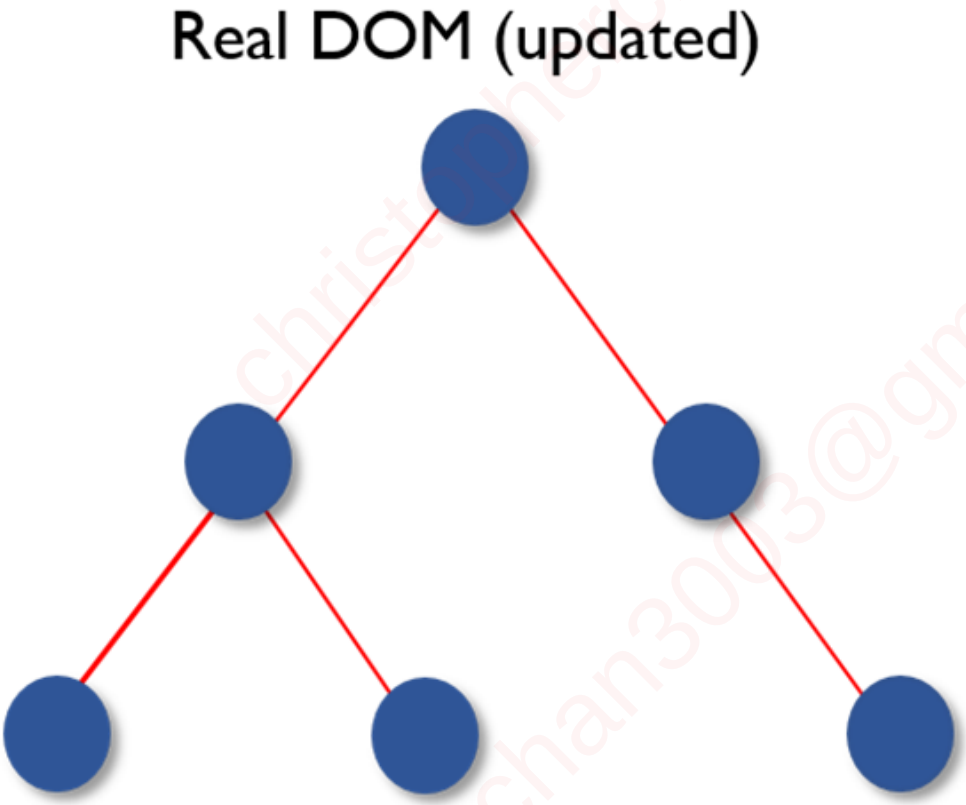
Virtual DOM 是一个轻量级的 JavaScript 对象，它最初只是 real DOM 的副本。它是一个节点树，它将元素、它们的属性和内容作为对象及其属性。React 的渲染函数从 React 组件中创建一个节点树。然后它响应数据模型中的变化来更新该树，该变化是由用户或系统完成的各种动作引起的。
Virtual DOM 工作过程有三个简单的步骤。

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。



Q48. 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:

```
1.  class MyComponent extends React.Component{
2.      render(){
3.          return(
4.              <div>
5.                  <h1>Hello</h1>
6.                  <Header/>
7.              </div>
8.          );
9.      }
10. }
11. class Header extends React.Component{
12.     render(){
13.         return
14.             <h1>Header Component</h1>
15.     };
16. }
17. ReactDOM.render(
18.     <MyComponent/>, document.getElementById('content')
19. );
```

Q49. 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。

面试题

Q50. 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。

Q51. 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

Q52. 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')
);
```

Q53. 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。

```
1.  //General way
2.  render() {
3.      return(
4.          <MyInput onChange = {this.handleChange.bind(this) } />
5.      );
6.  }
7.  //With Arrow Function
8.  render() {
9.      return(
10.         <MyInput onChange = { (e)=>this.handleOnChange(e) } />
11.     );
12. }
```

Q54. 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。

面试题

Q55. 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：
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。

Q56. 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>
        );
    }
});
```

Q57. 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。

Q58. 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>
    );
  }
}
```

Q59. 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

Q60. 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。

Q61. 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

面试题

Q62. 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。

```
1.  //ChildComponent.jsx
2.  export default class ChildComponent extends React.Component {
3.      render() {
4.          return(
5.              <div>
6.                <h1>This is a child component</h1>
7.              </div>
8.          );
9.      }
10. }
11.
12. //ParentComponent.jsx
13. import ChildComponent from './childcomponent.js';
14. class ParentComponent extends React.Component {
15.     render() {
16.         return(
17.             <div>
18.               <App />
19.             </div>
20.         );
21.     }
22. }
```