FullStack.Cafe - Never Fail Tech Interview (https://www.fullstack.cafe)

39 Advanced React Interview Questions You Must Clarify (2020 Update)

Originally published on 39 Advanced React Interview Questions You Must Clarify (2020 Update) | FullStack.Cafe (https://www.fullstack.cafe\blog\react-js-interview-questions)

- Q1 : What is virtual DOM?
- Q2 : What are the differences between a class component and functional component?
- Q3 : What are refs used for in React?
- Q4 : Describe how events are handled in React.
- Q5 : What is the difference between state and props?
- Q6 : How to create refs?
- Q7 : What are Higher-Order components?
- Q8 : What is the purpose of using super constructor with props argument?
- Q9 : What are controlled components?
- Q10 : What is equivalent of the following using React.createElement?
- Q11 : What can you tell me about JSX?
- Q12 : Given the code defined above, can you identify two problems?
- Q13 : Why should not we update the state directly?
- Q14 : What are the different phases of ReactJS component lifecycle?
- Q15 : What are the lifecycle methods of ReactJS?
- Q16 : What do these three dots (...) in React do?

What does the ... do in this React (using JSX) code and what is it called?

```
<Modal {...this.props} title='Modal heading' animation={fal
```

- Q17 : What are advantages of using React Hooks?
- Q18 : What are React Hooks?
- Q19 : What is useState() in React?

Explain what is the use of useState(0) there:

```
...
const [count, setCounter] = useState(0);
const [moreStuff, setMoreStuff] = useState(...);
...

const setCount = () => {
    setCounter(count + 1);
    setMoreStuff(...);
    ...
};
```

- Q20 : What is StrictMode in React?
- Q21 : Why do class methods need to be bound to a class instance?
- Q22 : What is prop drilling and how can you avoid it?
- Q23 : Describe Flux vs MVC?
- Q24 : What is the difference between a controlled component and an uncontrolled component?
- Q25 : What is wrong with this code?
- Q26 : What is the React context?
- Q27 : What is React Fiber?
- Q28 : How to apply validation on Props in ReactJS?
- Q29 : What is the difference between ReactJS and Angular?
- Q30 : What is the difference between using constructor vs getInitialState in React?
- Q31 : When is it important to pass props to super(), and why?
- Q32 : How to conditionally add attributes to React components?

Is there a way to only add attributes to a React component if a certain condition is met?

- Q33 : Do Hooks replace render props and higher-order components?
- Q34 : How would you go about investigating slow React application rendering?
- Q35 : When would you use StrictMode component in React?
- Q36 : What is a pure function?
- Q37 : How does React renderer work exactly when we call setState?
- Q38 : What is the key architectural difference between a JavaScript library such as React and a JavaScript framework such as Angular?
- Q39 : How to avoid the need for binding in React?

Answers

Q1: What is virtual DOM? ★

Topic: React

The virtual DOM (VDOM) is an in-memory representation of Real DOM. The representation of a UI is kept in memory and synced with the “real” DOM. It’s a step that happens between the render function being called and the displaying of elements on the screen. This entire process is called reconciliation.

Q2: What are the differences between a class component and functional component? ★★

Topic: React

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

Q3: What are refs used for in React? ★★

Topic: React

Refs are an escape hatch which allow you to get direct access to a DOM element or an instance of a component. In order to use them you add a ref attribute to your component whose value is a callback function which will receive the underlying DOM element or the mounted instance of the component as its first argument.

```
class UnControlledForm extends Component {
  handleSubmit = () => {
    console.log("Input Value: ", this.input.value)
  }
  render () {
    return (
      <form onSubmit={this.handleSubmit}>
        <input
          type='text'
          ref={(input) => this.input = input} />
        <button type='submit'>Submit</button>
      </form>
    )
  }
}
```

Above notice that our input field has a ref attribute whose value is a function. That function receives the actual DOM element of input which we then put on the instance in order to have access to it inside of the handleSubmit function.

It’s often misconstrued that you need to use a class component in order to use refs, but refs can also be used with functional components by leveraging closures in JavaScript.

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

Q4: Describe how events are handled in React. ★★

Topic: React

In order to solve cross browser compatibility issues, your event handlers in React will be passed instances of SyntheticEvent, which is React's cross-browser wrapper around the browser's native event. These synthetic events have the same interface as native events you're used to, except they work identically across all browsers.

What's mildly interesting is that React doesn't actually attach events to the child nodes themselves. React will listen to all events at the top level using a single event listener. This is good for performance and it also means that React doesn't need to worry about keeping track of event listeners when updating the DOM.

Q5: What is the difference between state and props? ★★

Topic: React

Both props and state are plain JavaScript objects. While both of them hold information that influences the output of render, they are different in their functionality with respect to component. i.e,

- Props get passed to the component similar to function parameters
- state is managed within the component similar to variables declared within a function.

Q6: How to create refs? ★★

Topic: React

Refs are created using React.createRef() method and attached to React elements via the ref attribute. In order to use refs throughout the component, just assign the ref to the instance property with in constructor.

```
class MyComponent extends React.Component {
  constructor(props) {
    super(props);
    this.myRef = React.createRef();
  }
  render() {
    return <div ref={this.myRef} />;
  }
}
```

And:

```
class UserForm extends Component {
 handleSubmit = () => {
  console.log("Input Value is: ", this.input.value)
 }
 render () {
  return (
   <form onSubmit={this.handleSubmit}>
    <input
     type='text'
     ref={(input) => this.input = input} /> // Access DOM input in handle submit
    <button type='submit'>Submit</button>
   </form>
  )
 }
}
```

We can also use it in functional components with the help of closures.

Q7: What are Higher-Order components? ★★

Topic: React

A higher-order component (HOC) is a function that takes a component and returns a new component. Basically, it’s a pattern that is derived from React’s compositional nature
We call them as “pure’ components” because they can accept any dynamically provided child component but they won’t modify or copy any behavior from their input components.

```
const EnhancedComponent = higherOrderComponent(WrappedComponent);
```

HOC can be used for many use cases as below,

1. Code reuse, logic and bootstrap abstraction
2. Render High jacking
3. State abstraction and manipulation
4. Props manipulation

Q8: What is the purpose of using super constructor with props argument? ★★

Topic: React

A child class constructor cannot make use of this reference until super() method has been called. The same applies for ES6 sub-classes as well. The main reason of passing props parameter to super() call is to access this.props in your child constructors.

Passing props:

```
class MyComponent extends React.Component {
   constructor(props) {
      super(props);
      console.log(this.props);  // Prints { name: 'sudheer',age: 30 }
   }
}
```

Not passing props:

```
class MyComponent extends React.Component {
  constructor(props) {
    super();
    console.log(this.props); // Prints undefined
    // But Props parameter is still available
    console.log(props); // Prints { name: 'sudheer',age: 30 }
  }

  render() {
    // No difference outside constructor
    console.log(this.props) // Prints { name: 'sudheer',age: 30 }
  }
}
```

The above code snippets reveals that this.props behavior is different only with in the constructor. It would be same outside the constructor.

Q9: What are controlled components? ★★★

Topic: React

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. An input form element whose value is controlled by React in this way is called a controlled component.

Q10: What is equivalent of the following using React.createElement? ★★★

Topic: React

Question:

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

What is equivalent of the following using React.createElement?

Answer:

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

Q11: What can you tell me about JSX? ★★★

Topic: React

When Facebook first released React to the world, they also introduced a new dialect of JavaScript called JSX that embeds raw HTML templates inside JavaScript code. JSX code by itself cannot be read by the browser; it must be transpiled into traditional JavaScript using tools like Babel and webpack. While many developers understandably have initial knee-jerk reactions against it, JSX (in tandem with ES2015) has become the defacto method of defining React components.

```
class MyComponent extends React.Component {
  render() {
    let props = this.props;
    return (
      <div className="my-component">
      <a href={props.url}>{props.name}</a>
      </div>
    );
  }
}
```

Q12: Given the code defined above, can you identify two problems? ★★★

Topic: React

Take a look at the code below:

```
class MyComponent extends React.Component {
  constructor(props) {
    // set the default internal state
    this.state = {
      clicks: 0
    };
  }

  componentDidMount() {
    this.refs.myComponentDiv.addEventListener('click', this.clickHandler);
  }

  componentWillUnmount() {
    this.refs.myComponentDiv.removeEventListener('click', this.clickHandler);
  }

  clickHandler() {
    this.setState({
      clicks: this.clicks + 1
    });
  }

  render() {
    let children = this.props.children;

    return (
      <div className="my-component" ref="myComponentDiv">
      <h2>My Component ({this.state.clicks} clicks})</h2>
      <h3>{this.props.headerText}</h3>
    {children}
    </div>
    );
  }
}
```

Given the code defined above, can you identify two problems?

Answer:

1. The constructor does not pass its props to the super class. It should include the following line:

```
constructor(props) {
  super(props);
  // ...
}
```

2. The event listener (when assigned via addEventListener()) is not properly scoped because ES2015 doesn't provide autobinding (https://facebook.github.io/react/docs/reusable-components.html#no-autobinding). Therefore the developer can re-assign clickHandler in the constructor to include the correct binding to this:

```
constructor(props) {
  super(props);
  this.clickHandler = this.clickHandler.bind(this);
  // ...
}
```

Q13: Why should not we update the state directly? ★★★

Topic: React

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’});
```

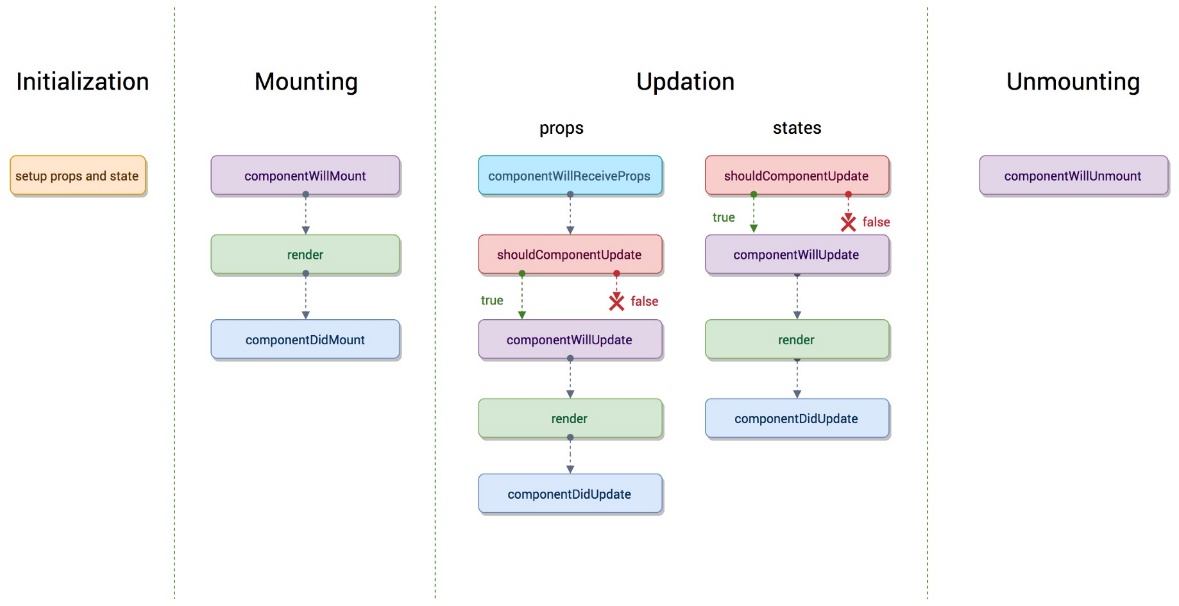
Note: The only place you can assign the state is constructor.

Q14: What are the different phases of ReactJS component lifecycle? ★★★

Topic: React

There are four different phases of React component’s lifecycle:

1. Initialization: In this phase react component prepares setting up the initial state and default props.
2. Mounting: The react component is ready to mount in the browser DOM. This phase covers componentWillMount and componentDidMount lifecycle methods.
3. Updating: In this phase, the component get updated in two ways, sending the new props and updating the state. This phase covers shouldComponentUpdate, componentWillUpdate and componentDidUpdate lifecycle methods.
4. Unmounting: In this last phase, the component is not needed and get unmounted from the browser DOM. This phase include componentWillUnmount lifecycle method.



Q15: What are the lifecycle methods of ReactJS? ★★★

Topic: React

- 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 eventListeners 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 rerender if component receives new prop.
- componentWillUpdate: Executed before re-rendering the component when there are pros & 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.

Q16: What do these three dots (...) in React do? ★★★

Topic: React

Details:
What does the ... do in this React (using JSX) code and what is it called?

```
<Modal {...this.props} title='Modal heading' animation={fal
```

Answer:

That's property spread notation. It was added in ES2018 (spread for arrays/iterables was earlier, ES2015).

For instance, if this.props contained a: 1 and b: 2, then

```
<Modal {...this.props} title='Modal heading' animation={false}>
```

would be the same as:

```
<Modal a={this.props.a} b={this.props.b} title='Modal heading' animation={false}>
```

Spread notation is handy not only for that use case, but for creating a new object with most (or all) of the properties of an existing object — which comes up a lot when you're updating state, since you can't modify state directly:

```
this.setState(prevState => {
   return {foo: {...prevState.foo, a: "updated"}};
});
```

Q17: What are advantages of using React Hooks? ★★★

Topic: React

Primarily, hooks in general enable the extraction and reuse of stateful logic that is common across multiple components without the burden of higher order components or render props. Hooks allow to easily manipulate the state of our functional component without needing to convert them into class components.

Hooks don’t work inside classes (because they let you use React without classes). By using them, we can totally avoid using lifecycle methods, such as componentDidMount, componentDidUpdate, componentWillUnmount. Instead, we will use built-in hooks like useEffect .

Q18: What are React Hooks? ★★★

Topic: React

Hooks are a new addition in React 16.8. They let you use state and other React features without writing a class. With Hooks, you can extract stateful logic from a component so it can be tested independently and reused. Hooks allow you to reuse stateful logic without changing your component hierarchy. This makes it easy to share Hooks among many components or with the community.

Q19: What is useState() in React? ★★★

Topic: React

Details:
Explain what is the use of useState(0) there:

```
...
const [count, setCounter] = useState(0);
const [moreStuff, setMoreStuff] = useState(...);
...

const setCount = () => {
   setCounter(count + 1);
   setMoreStuff(...);
   ...
};
```

Answer:

useState is one of build-in react hooks. useState(0) returns a tuple where the first parameter count is the current state of the counter and setCounter is the method that will allow us to update the counter's state.

We can use the setCounter method to update the state of count anywhere - In this case we are using it inside of the setCount function where we can do more things; the idea with hooks is that we are able to keep our code more functional and avoid class based components if not desired/needed.

Q20: What is StrictMode in React? ★★★

Topic: React

React's StrictMode is sort of a helper component that will help you write better react components, you can wrap a set of components with <StrictMode /> and it'll basically:

- Verify that the components inside are following some of the recommended practices and warn you if not in the console.
- Verify the deprecated methods are not being used, and if they're used strict mode will warn you in

the console.
- Help you prevent some side effects by identifying potential risks.

Q21: Why do class methods need to be bound to a class instance? ★★★

Topic: React

In JavaScript, the value of this changes depending on the current context. Within React class component methods, developers normally expect this to refer to the current instance of a component, so it is necessary to *bind* these methods to the instance. Normally this is done in the constructor—for example:

```
class SubmitButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      isFormSubmitted: false
    };
    this.handleSubmit = this.handleSubmit.bind(this);
  }

  handleSubmit() {
    this.setState({
      isFormSubmitted: true
    });
  }

  render() {
    return (
      <button onClick={this.handleSubmit}>Submit</button>
    )
  }
}
```

Q22: What is prop drilling and how can you avoid it? ★★★

Topic: React

When building a React application, there is often the need for a deeply nested component to use data provided by another component that is much higher in the hierarchy. The simplest approach is to simply pass a prop from each component to the next in the hierarchy from the source component to the deeply nested component. This is called prop drilling.

The primary disadvantage of prop drilling is that components that should not otherwise be aware of the data become unnecessarily complicated and are harder to maintain.

To avoid prop drilling, a common approach is to use React context. This allows a Provider component that supplies data to be defined, and allows nested components to consume context data via either a Consumer component or a useContext hook.

Q23: Describe Flux vs MVC? ★★★★

Topic: React

Read Full Answer on [FullStack.Cafe (https://www.fullstack.cafe\blog\react-js-interview-questions)](https://www.fullstack.cafe\blog\react-js-interview-questions)

Q24: What is the difference between a controlled component and an uncontrolled component? ★★★★

Topic: React

Read Full Answer on [FullStack.Cafe (https://www.fullstack.cafe\blog\react-js-interview-questions)](https://www.fullstack.cafe/blog/react-js-interview-questions)

Q25: What is wrong with this code? ★★★★

Topic: React

Read Full Answer on [FullStack.Cafe (https://www.fullstack.cafe\blog\react-js-interview-questions)](https://www.fullstack.cafe/blog/react-js-interview-questions)

Q26: What is the React context? ★★★★

Topic: React

Read Full Answer on [FullStack.Cafe (https://www.fullstack.cafe\blog\react-js-interview-questions)](https://www.fullstack.cafe/blog/react-js-interview-questions)

Q27: What is React Fiber? ★★★★

Topic: React

Read Full Answer on [FullStack.Cafe (https://www.fullstack.cafe\blog\react-js-interview-questions)](https://www.fullstack.cafe/blog/react-js-interview-questions)

Q28: How to apply validation on Props in ReactJS? ★★★★

Topic: React

Read Full Answer on [FullStack.Cafe (https://www.fullstack.cafe\blog\react-js-interview-questions)](https://www.fullstack.cafe/blog/react-js-interview-questions)

Q29: What is the difference between ReactJS and Angular? ★★★★

Topic: React

Read Full Answer on [FullStack.Cafe (https://www.fullstack.cafe\blog\react-js-interview-questions)](https://www.fullstack.cafe/blog/react-js-interview-questions)

Q30: What is the difference between using constructor vs getInitialState in React? ★★★★

Topic: React

Read Full Answer on [FullStack.Cafe (https://www.fullstack.cafe\blog\react-js-interview-questions)](https://www.fullstack.cafe/blog/react-js-interview-questions)

Q31: When is it important to pass props to super(), and why? ★★★★

Topic: React

Read Full Answer on [FullStack.Cafe (https://www.fullstack.cafe\blog\react-js-interview-questions)](https://www.fullstack.cafe/blog/react-js-interview-questions)

Q32: How to conditionally add attributes to React components? ★★★★

Topic: React

Details:
Is there a way to only add attributes to a React component if a certain condition is met?

Answer:

Read Full Answer on [FullStack.Cafe (https://www.fullstack.cafe\blog\react-js-interview-questions)](https://www.fullstack.cafe/blog/react-js-interview-questions)

Q33: Do Hooks replace render props and higher-order components? ★★★★

Topic: React

Read Full Answer on [FullStack.Cafe (https://www.fullstack.cafe\blog\react-js-interview-questions)](https://www.fullstack.cafe/blog/react-js-interview-questions)

Q34: How would you go about investigating slow React application rendering? ★★★★

Topic: React

Read Full Answer on [FullStack.Cafe (https://www.fullstack.cafe\blog\react-js-interview-questions)](https://www.fullstack.cafe/blog/react-js-interview-questions)

Q35: When would you use StrictMode component in React? ★★★★

Topic: React

Read Full Answer on [FullStack.Cafe (https://www.fullstack.cafe\blog\react-js-interview-questions)](https://www.fullstack.cafe/blog/react-js-interview-questions)

Q36: What is a pure function? ★★★★★

Topic: React

Read Full Answer on [FullStack.Cafe (https://www.fullstack.cafe\blog\react-js-interview-questions)](https://www.fullstack.cafe/blog/react-js-interview-questions)

Q37: How does React renderer work exactly when we call setState? ★★★★★

Topic: React

Read Full Answer on [FullStack.Cafe (https://www.fullstack.cafe\blog\react-js-interview-questions)](https://www.fullstack.cafe/blog/react-js-interview-questions)

Q38: What is the key architectural difference between a JavaScript library such as React and a JavaScript framework such as Angular? ★★★★★

Topic: React

Read Full Answer on [FullStack.Cafe (https://www.fullstack.cafe\blog\react-js-interview-questions)](https://www.fullstack.cafe/blog/react-js-interview-questions)

Q39: How to avoid the need for binding in React? ★★★★★

Topic: React

Read Full Answer on [FullStack.Cafe (https://www.fullstack.cafe\blog\react-js-interview-questions)](https://www.fullstack.cafe/blog/react-js-interview-questions)