**Must do interview questions:**

1. **React Life Cycle hooks**
2. **CSS learn types of selectors**
3. **Learn Promises, callbacks, async/await, observables**
4. **Learn scoing, var, let and const hoising**
5. **White Board, string manipulation, palindromes, fibonacci**

**What is React?**

React is an open source front end JavaScript library used for building interactive single page web applications. It was developed by Jordan Walke in 2011.

**What are the features of React?**

The main features of React are as follows:

1. It uses virtual DOM instead of Real DOM. They are cheap to create.
2. It uses server-side rendering.
3. It uses reusable/composable components
4. It follows unidirectional data flow

**What is JSX?**

JSX are XML like syntax extension to ECMA script. They are like JavaScript functions. We can pass them to a function, return them from function, can also be used as a value of a const.

*const x = <h1>Zero</h1>*

What is an Element?

Elements are JavaScript objects describing what is going to be rendered onto the screen. React element has four properties: type, props, key and ref. It’s just a plain JavaScript object describing a DOM node or a component and its desired properties, along with the composition using props.children.

What is a component in React?

Every react app is essentially a **tree of components**. At the heart of all the react app are the **COMPONENTS** which is a piece of user interface. Simply put, a component is a **JavaScript class or function** that optionally accepts inputs i.e. properties(props) and returns a **React** element that describes how a section of the UI (User Interface) should appear. In terms of **implementation**, a component is implemented as a **JavaScript class** that has **some state and render method**. When building react apps, we build a bunch of **independent, isolated, and reusable** react components which then is **combined** to build complex user interfaces. Every react app has the **root component** which we call the **App component**.

**How to create components in React?**

There are two possible ways to create a component in React.

1. Functional Components: It is the easiest way to create a component. It is a pure JavaScript function that accepts props objects as first parameter and return react elements.
2. Class components: We can also use ES6 class to define a component.

**When to use class components over a function component?**

If the component needs a state or lifecycle methods then we use class component otherwise we can use functional component. However, from React 16.8 with the addition of Hooks, you could use state , lifecycle methods and other features that were only available in class component right in your function component.

**What are pure components?**

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

### What is state in React?

***The state of a component is an object*** ***that holds some data that we want to display when the component (part or piece) is rendered (provide)***. The data may change over the lifetime of the component. We should always try to make our state as simple as possible and minimize the number of stateful components.

### What are props in React?

Every react components has a property called props and it is basically a plain JavaScript object that includes all the attributes that we set. So in the counters.jsx, value and selected are the properties of the props. They are data passed down from a parent component to a child component.

The primary purpose of props in React is to provide following component functionality:

1. Pass custom data to your component.
2. Trigger state changes.
3. Use via this.props.reactProp inside component's render() method.

**What is the difference between state and props?**

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

**Why should we not update the state directly?**

If you try to update state directly then it won't re-render the component. Instead use setState() method. It schedules an update to a component's state object. When state changes, the component responds by re-rendering.

**What is the purpose of callback function as an argument of setState()?**

The callback function is invoked when setState finished and the component gets rendered. Since setState() is asynchronous it means the value of the state does not immediately change after calling this function. The value does not get automatically updated. In order to update the state right away we need to use a function called callback().

Note: It is recommended to use lifecycle method componentDidUpdate rather than this callback function.

**When not to use the callbacks?**

The docs recommend that you use the lifecycle events.

Here’s why.

[PureComponent](https://facebook.github.io/react/docs/react-api.html#react.purecomponent) and [shouldComponentUpdate](https://facebook.github.io/react/docs/react-component.html#shouldcomponentupdate) can be used to tune up a component’s performance. They work by preventing lifecycle methods from firing when props and state haven’t changed. The setState callback fires regardless of what shouldComponentUpdate returns. So, the setState callback will fire, even when state hasn’t changed. Don’t be afraid to use the setState callback. It’s there for a reason. But when you do, keep an eye on shouldComponentUpdate, if you see any shiftiness.

### What is the difference between HTML and React event handling?

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

Whereas in React it follows *camelCase* convention:

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

Whereas in React you must call preventDefault() explicitly:

**How to bind methods or event handlers in JSX callbacks?**

There are 3 possible ways to achieve this:

Binding in Constructor: In JavaScript classes, the methods are not bound by default. The same thing applies for React event handlers defined as class methods. **Normally we bind them in constructor.**

class Component extends React.Componenet {

constructor(props) {

super(props)

this.handleClick = this.handleClick.bind(this)

}

handleClick() {

// ...

}

}

**Public class fields syntax**: If you don't like to use bind approach then public class fields syntax can be used to correctly bind callbacks.

handleClick = () => {

console.log('this is:', this)

}

<button onClick={this.handleClick}>

{'Click me'}

</button>

**Arrow functions in callbacks:** You can use arrow functions directly in the callbacks.

<button onClick={(event) => this.handleClick(event)}>

{'Click me'}

</button>

Note: If the callback is passed as prop to child components, those components might do an extra re-rendering. In those cases, it is preferred to go with .bind() or public class fields syntax approach considering performance.

### How to pass a parameter to an event handler or callback?

You can use an arrow function to wrap around an event handler and pass parameters.

<button onClick={() => this.handleClick(id)} />

This is an equivalent to calling .bind:

<button onClick={this.handleClick.bind(this, id)} />

### What are synthetic events in React?

SyntheticEvent is a cross-browser wrapper around the browser's native event. It's API is same as the browser's native event, including stopPropagation() and preventDefault(), except the events work identically across all browsers.

### What is inline conditional expressions?

You can use either if statements or ternary expressions which are available from JS to conditionally render expressions. Apart from these approaches, you can also embed any expressions in JSX by wrapping them in curly braces and then followed by JS logical operator &&.

<h1>Hello!</h1>

{

messages.length > 0 && !isLogin?

<h2>

You have {messages.length} unread messages.

</h2>

:

<h2>

You don't have unread messages.

</h2>

}

### What are "key" props and what is the benefit of using them in arrays of elements?

A key is a special string attribute you **should** include when creating arrays of elements. Keys help React identify which items have changed, are added, or are removed.

### What is the use of refs?

The ref is used to return a reference to the element. They should be avoided in most cases, however, they can be useful when you need a direct access to the DOM element or an instance of a component.