1. What is React?

React is a JavaScript library used for building user interfaces in web applications. It's maintained by Facebook and a community of developers. What makes React special is its component-based architecture, which allows developers to create reusable UI elements and manage state more efficiently. React is known for its performance optimizations, such as the virtual DOM, which helps in creating fast and responsive applications.

2. What are the major features of React?

React offers several key features:

- Virtual DOM: React uses a virtual representation of the DOM to minimize the number of actual DOM manipulations, resulting in improved performance.

- JSX: JSX is a syntax extension for JavaScript that allows developers to write HTML-like code within their JavaScript files, making it easier to create and visualize UI components.

- Components: React applications are built using components, which are reusable and composable pieces of UI. Components encapsulate both the UI and its behavior, making code organization and maintenance easier.

- One-way Data Flow: React follows a unidirectional data flow pattern, where data flows from parent to child components. This ensures predictable behavior and makes it easier to debug applications.

3. What is JSX?

JSX, or JavaScript XML, is a syntax extension for JavaScript that allows developers to write HTML-like code within their JavaScript files. JSX makes it easier to create and visualize UI components by combining JavaScript logic with markup. It's not mandatory to use JSX in React, but it's a popular choice because of its readability and familiarity to developers.

4. What is the difference between Element and Component?

In React, an Element is a plain JavaScript object that represents a single instance of a UI element. It describes what you want to see on the screen. A Component, on the other hand, is a JavaScript function or class that returns a React element. Components can be thought of as building blocks for UI, composed of one or more elements.

5. How to create components in React?

Components in React can be created using either classes or functions. Class components are JavaScript classes that extend the `React.Component` class and define a `render()` method to return the desired JSX. Functional components are simpler and more concise, as they are JavaScript functions that return JSX directly.

6. When to use a Class Component over a Function Component?

Class components should be used when you need to manage state, use lifecycle methods, or optimize performance by implementing methods like `shouldComponentUpdate()`. Function components are preferred for simpler presentational components or when utilizing React hooks for state management and side effects.

7. What are Pure Components?

Pure Components are a type of class component in React that automatically optimize rendering performance by implementing a shallow comparison check in the `shouldComponentUpdate()` method. This comparison helps prevent unnecessary re-renders when the props or state of a component haven't changed.

8. What is state in React?

State in React is an object that represents the current condition of a component. It holds information that can change over time due to user interactions, network responses, or other factors. By using state, components can manage their own data and trigger UI updates when the state changes.

9. What are props in React?

Props, short for properties, are a way to pass data from a parent component to a child component in React. They are immutable and are used to customize the behavior and appearance of a component. Props allow components to be reusable and configurable, as they can accept different sets of data.

10. What is the difference between state and props?

The main difference between state and props in React is where the data comes from and whether it can be changed. State is managed within a component and can be changed over time, while props are passed from parent to child components and remain immutable within the component receiving them.

11. Why should we not update the state directly?

Directly mutating state in React can lead to unexpected behavior and bugs in your application. React's `setState()` method should be used to update state because it ensures that React can properly track state changes and trigger re-renders as needed.

12. What is the purpose of a callback function as an argument of setState()?

The callback function passed to the `setState()` method in React is executed after the state has been updated and the component has been re-rendered. It's useful for performing additional tasks or accessing the updated state after a state update operation.

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

In React, event handling is similar to HTML event handling, but with some differences. Event names are written in camelCase instead of lowercase, and event handlers are passed as functions rather than strings. This allows React to manage event binding and provides a more consistent interface for handling events.

14. How to bind methods or event handlers in JSX callbacks?

In React, methods or event handlers can be bound to components using arrow functions or by using the `bind()` method in the component's constructor. Arrow functions automatically bind the current context (`this`), while `bind()` explicitly binds a method to a specific context.

15. How to pass a parameter to an event handler or callback?

To pass parameters to event handlers or callbacks in React, you can use arrow functions or the `bind()` method to create a new function that includes the additional parameters. This allows you to access the parameters within the event handler when it's invoked.

16. What are synthetic events in React?

Synthetic events in React are cross-browser wrappers around native browser events. They provide a consistent interface for handling events across different browsers and ensure that event handling behaves predictably within React applications.

17. What are inline conditional expressions?

Inline conditional expressions in React allow you to conditionally render content within JSX using JavaScript expressions. This enables dynamic rendering based on certain conditions without the need for if-else statements, making your code more concise and readable.

18. What is the "key" prop and what is the benefit of using it in arrays of elements?

The "key" prop is a special attribute in React that helps identify which items in a list have changed, been added, or been removed. It improves performance by enabling React to efficiently update the UI when working with dynamic lists of elements, such as those generated from arrays.

19. What is the use of refs?

Refs in React provide a way to access and interact with DOM elements or React components directly. They are useful for scenarios such as focusing on an input field, triggering imperative animations, or integrating with third-party libraries that require direct DOM access.

20. How to create refs?

Refs in React can be created using the `React.createRef()` method or the `useRef()` hook. This creates a reference object that can be attached to a DOM element or React component. Once created, refs can be accessed and used to perform actions such as focusing, measuring, or animating elements.

```
import React, { useRef } from 'react';

function MyComponent() {

  const inputRef = useRef(null);

  const handleClick = () => {

    inputRef.current.focus();

  };

  return (

    <div>

      <input ref={inputRef} type="text" />

      <button onClick={handleClick}>Focus Input</button>

    </div>

  );

}
```

21. What are forward refs?

Forward refs are a feature in React that allow components to pass refs to their children components. This is useful when you need to access the DOM nodes or React elements of child components from a parent component.

22. Which is the preferred option within callback refs and findDOMNode()?

The preferred option is callback refs. Callback refs provide a more flexible and reliable way to access DOM nodes or React elements compared to `findDOMNode()`. `findDOMNode()` is considered legacy and should be avoided in most cases due to potential performance issues and compatibility concerns.

23. Why are String Refs legacy?

String refs (refs using strings) are considered legacy because they have several drawbacks compared to callback refs. They are less flexible, harder to manage, and can cause issues with code minification. Callback refs offer a better alternative for accessing and managing refs in React components.

24. What is Virtual DOM?

The Virtual DOM is a concept in React that represents a lightweight copy of the actual DOM (Document Object Model) tree. It's a JavaScript representation of the UI hierarchy, allowing React to perform efficient updates by comparing changes in the virtual DOM and then applying only the necessary changes to the real DOM.

25. How Virtual DOM works?

When changes are made to a React component, React first updates the virtual DOM instead of the real DOM. It then compares the updated virtual DOM with the previous version to determine the minimal set of changes needed to update the real DOM. Finally, React applies these changes to the real DOM, resulting in efficient and optimized UI updates.

26. What is the difference between Shadow DOM and Virtual DOM?

Shadow DOM is a web standard that allows encapsulation of DOM and CSS in a component, while Virtual DOM is a concept specific to React for optimizing DOM updates. Shadow DOM is used for encapsulating styles and markup within a component, while Virtual DOM is used for efficient updating of the DOM tree.

27. What is React Fiber?

React Fiber is a reimplementation of the React reconciliation algorithm, which is responsible for updating the UI in response to changes in data or state. Fiber is designed to be more efficient and incremental, allowing React to prioritize and schedule updates in a way that improves performance and responsiveness.

28. What is the main goal of React Fiber?

The main goal of React Fiber is to improve the performance and responsiveness of React applications, especially for complex and interactive user interfaces. Fiber achieves this by introducing a more efficient reconciliation algorithm and implementing features like incremental rendering and prioritized updates.

29. What are controlled components?

Controlled components are React components whose behavior is controlled entirely by React state. This means that the value of the component (such as an input field) is controlled by React state, and any changes to the value are handled by React, allowing for precise control and synchronization with other components or data.

30. What are uncontrolled components?

Uncontrolled components are React components whose behavior is not entirely controlled by React state. Instead, they rely on the DOM to manage their state, such as form inputs that maintain their own state internally. Uncontrolled components are useful when you need a simpler way to manage state or integrate with non-React libraries.

31. What is the difference between createElement and cloneElement?

`createElement` is a function used to create React elements directly, while `cloneElement` is a function used to clone and modify existing React elements. `createElement` is typically used to create new elements from scratch, while `cloneElement` is used to modify existing elements with new props.

32. What is Lifting State Up in React?

Lifting State Up is a pattern in React where the state of a component is moved up to a common ancestor component. This allows multiple child components to share the same state and synchronize their behavior, making it easier to manage and update state across the application.

33. What are the different phases of component lifecycle?

The component lifecycle in React consists of three main phases:

- Mounting: When a component is first created and inserted into the DOM.
- Updating: When a component's props or state change, causing it to re-render.
- Unmounting: When a component is removed from the DOM.

34. What are the lifecycle methods of React?

React components have several lifecycle methods that allow developers to hook into different stages of a component's lifecycle. Some common lifecycle methods include `componentDidMount`, `componentDidUpdate`, and `componentWillUnmount`.

35. What are Higher-Order components?

Higher-Order components (HOCs) are a pattern in React that involves wrapping a component with another component to add or modify its behavior. HOCs are used for code reuse, logic sharing, and abstraction of complex behavior.

36. How to create props proxy for HOC component?

A props proxy is a technique used in Higher-Order components to pass props down to the wrapped component. This involves creating a new component that renders the original component with additional props or modified behavior.

37. What is context?

Context is a feature in React that allows data to be passed down through the component tree without having to manually pass props at each level. It's useful for sharing global data, such as themes or user preferences, across multiple components in an application.

38. What is children prop?

The `children` prop in React is a special prop that represents the content between the opening and closing tags of a component. It allows components to be used as containers for other components or content, making it easy to compose complex UIs.

39. How to write comments in React?

In React, comments can be written using JavaScript's single-line (`//`) or multi-line (`/* */`) comment syntax within JSX. Comments are ignored during the rendering process and do not appear in the final output.

40. What is the purpose of using super constructor with props argument?

In React class components, the `super(props)` constructor call is used to call the constructor of the parent class and pass the `props` object to it. This is necessary when defining a constructor in a subclass to ensure that React can properly initialize the component's props.

Sure, here are the answers to your questions:

41. What is reconciliation?

Reconciliation is the process in React where it compares the virtual DOM representation of a component with its previous state and determines the minimal set of changes needed to update the actual DOM. This process ensures that the UI remains in sync with the underlying data and state changes.

42. How to set state with a dynamic key name?

You can set state with a dynamic key name by using computed property names in JavaScript. Here's an example:

```javascript
const key = 'dynamicKey';

this.setState({ [key]: 'dynamicValue' });
```

43. What would be the common mistake of a function being called every time the component renders?

The common mistake would be to define a function directly within the `render()` method of a component. Doing so will cause the function to be recreated on every render, leading to unnecessary re-renders. To avoid this, functions should be defined outside the `render()` method or using class properties or hooks.

44. Is the lazy function supports named exports?

No, the `lazy` function in React does not directly support named exports. It is typically used with dynamic import() statements to lazily load components. However, you can still use named exports with lazy-loaded components by exporting them as default and then importing them with a different name.

45. Why does React use className over the class attribute?

React uses the `className` attribute instead of the `class` attribute to specify CSS classes for elements. This is because `class` is a reserved keyword in JavaScript, so using `className` avoids conflicts with JavaScript class syntax.

46. What are fragments?

Fragments are a way to group multiple React elements without adding an extra DOM node. They provide a cleaner way to structure JSX code by allowing components to return multiple elements without wrapping them in a parent div or other container.

47. Why are fragments better than container divs?

Fragments are better than container divs because they don't add unnecessary DOM nodes to the rendered output. This can lead to better performance and cleaner code, especially in cases where the extra container divs serve no purpose other than to satisfy the requirement of a single parent element.

48. What are portals in React?

Portals in React provide a way to render a component's children into a DOM node that exists outside of the parent component's DOM hierarchy. This allows components to render content outside of their usual position in the DOM tree, which can be useful for modals, tooltips, or other UI elements that need to be positioned relative to the document body.

49. What are stateless components?

Stateless components, also known as functional components, are components in React that are defined as plain JavaScript functions and do not have any internal state. They receive data via props and return JSX elements based on that data. Stateless components are simpler and easier to understand compared to class components.

50. What are stateful components?

Stateful components, also known as class components, are components in React that have an internal state managed by React. They are defined as ES6 classes and can hold and update their own data using the `setState()` method. Stateful components are useful for managing complex UI logic and data manipulation.

51. How to apply validation on props in React?

You can apply validation on props in React by using the `propTypes` property or by using TypeScript or Flow for static type checking. PropTypes allow you to define the expected types and shapes of props that a component should receive, while TypeScript and Flow provide static type checking capabilities for JavaScript code.

52. What are the advantages of React?

Some advantages of React include:

- Component-based architecture for better code organization and reusability.
- Virtual DOM for efficient and optimized rendering performance.
- Declarative syntax for easier development and debugging.
- Rich ecosystem with a large number of third-party libraries and tools.
- Strong community support and active development.

53. What are the limitations of React?

Some limitations of React include:

- Steeper learning curve for beginners, especially when using advanced features like hooks and context.
- JSX syntax might feel unfamiliar to developers coming from traditional HTML/CSS backgrounds.
- Limited built-in support for server-side rendering compared to other frameworks like Next.js or Nuxt.js.
- Large bundle sizes for complex applications, which can impact initial page load times.

54. What are error boundaries in React v16?

Error boundaries are a feature introduced in React v16 that allows components to catch JavaScript errors that occur during rendering, in lifecycle methods, and in constructors of the whole tree below them. They help prevent the entire React component tree from unmounting due to errors and provide a way to gracefully handle and display error messages to users.

55. How are error boundaries handled in React v15?

In React v15 and earlier versions, errors that occur during rendering or lifecycle methods would cause the entire React component tree to unmount, resulting in a blank screen or a broken UI. There was no built-in mechanism for handling errors gracefully, and developers had to rely on try-catch blocks or other error-handling strategies.

56. What are the recommended ways for static type checking?

Some recommended ways for static type checking in React include using TypeScript, Flow, or PropTypes. TypeScript and Flow are static type checkers for JavaScript that provide compile-time type checking, while PropTypes are a runtime type checker for React components that helps validate props during development.

57. What is the use of react-dom package?

The `react-dom` package is a package in React that provides DOM-specific methods for mounting and interacting with React components in web applications. It includes methods for rendering React components into the DOM, updating component props and state, and handling events.

58. What is the purpose of the render method of react-dom?

The `render` method of `react-dom` is used to render a React element into the DOM. It takes two arguments: the React element to render and the DOM node where the element should be rendered.

This method is typically called once in the entry point of a React application, such as the `index.js` file.

59. What is ReactDOMServer?

ReactDOMServer is a package in React that provides methods for rendering React components on the server side. It includes methods for rendering components to static markup (HTML) or to string, which can be used for server-side rendering, email generation, or other server-side rendering tasks.

60. How to use InnerHTML in React?

In React, dangerouslySetInnerHTML is used to set HTML directly from React components. It's generally discouraged to use this feature as it can expose your application to XSS (cross-site scripting) attacks if not used carefully. However, if you need to render HTML provided by a third-party source, you can use dangerouslySetInnerHTML with caution.

61. How to use styles in React?

Styles in React can be applied in various ways:

- Inline styles: Define styles directly within JSX using the `style` attribute.
- CSS Modules: Import CSS files as modules and apply styles using class names.
- CSS-in-JS libraries: Use libraries like styled-components or Emotion to define styles directly within JavaScript code.

62. How are events different in React?

Events in React are similar to regular DOM events but with some differences:

- Events are named using camelCase instead of lowercase.
- Event handlers are provided as attributes in JSX.
- Event handlers are automatically bound to the component's context, so you don't need to use `bind()`.

63. What will happen if you use setState in the constructor?

Using `setState()` in the constructor is not recommended because it can lead to unexpected behavior. React initializes the component's state before calling the constructor, so calling `setState()` directly in the constructor will overwrite the initial state with the values passed to `setState()`, causing inconsistencies.

64. What is the impact of indexes as keys?

Using indexes as keys for elements in a list can cause issues when the list order changes or items are added or removed. React uses keys to track elements and optimize updates, so using indexes can lead to incorrect rendering or performance issues, especially in dynamic lists.

65. Is it good to use setState() in componentWillMount() method?

No, it's not recommended to use `setState()` in the `componentWillMount()` method. This method is called before the initial render and is not recommended for side effects like updating state. Instead, you should use `componentDidMount()` for data fetching or state updates after the initial render.

66. What will happen if you use props in initial state?

Using props in the initial state of a component is not recommended because the state will not be updated if the props change later. React only sets the initial state once when the component is created, so changes to props will not be reflected in the state. Instead, you should use props directly in the render method or use `componentDidUpdate()` to update the state based on props changes.

67. How do you conditionally render components?

You can conditionally render components in React using JavaScript expressions within JSX. For example:

```jsx
{condition && <Component />}
```

This will render the `<Component />` only if the `condition` evaluates to true.

68. Why do we need to be careful when spreading props on DOM elements?

Spreading props on DOM elements can be dangerous because it may unintentionally pass invalid HTML attributes to the DOM, leading to unexpected behavior or security vulnerabilities. It's important to filter or sanitize props before spreading them to ensure that only valid attributes are passed to the DOM.

69. How do you use decorators in React?

Decorators in React are typically used with higher-order components (HOCs) to enhance or modify component behavior. You can use decorators by applying them to component classes or functions using the `@` syntax. For example:

```javascript
@withStyles(styles)

class MyComponent extends React.Component {
  // Component logic here
}
```

70. How do you memoize a component?

You can memoize a component in React using the `React.memo()` higher-order component. This function memoizes the rendered output of the component and re-renders only if its props change. For example:

```javascript
const MemoizedComponent = React.memo(MyComponent);
```

71. How do you implement Server-Side Rendering or SSR?

Server-Side Rendering (SSR) in React can be implemented using frameworks like Next.js or by setting up a custom server using Node.js and Express. SSR involves rendering React components on the server and sending the generated HTML to the client, which improves initial page load performance and SEO.

72. How to enable production mode in React?

To enable production mode in React, set the `NODE_ENV` environment variable to `'production'` before building or running the application. This will enable production optimizations like minification, dead code elimination, and performance improvements.

73. What is CRA and its benefits?

CRA stands for Create React App, which is a popular toolchain for creating React applications with no build configuration. CRA provides a pre-configured setup with tools like Webpack, Babel, and ESLint, allowing developers to focus on writing code rather than configuring build tools.

74. What is the lifecycle methods order in mounting?

In mounting phase, the lifecycle methods are called in the following order:

1. `constructor()`
2. `static getDerivedStateFromProps()`
3. `render()`
4. `componentDidMount()`

75. What are the lifecycle methods going to be deprecated in React v16?

In React v16, the following lifecycle methods are going to be deprecated:

- `componentWillMount()`
- `componentWillReceiveProps()`
- `componentWillUpdate()`

76. What is the purpose of getDerivedStateFromProps() lifecycle method?

The `getDerivedStateFromProps()` method is a static lifecycle method in React that is invoked right before rendering when new props or state are being received. It returns an object to update the state or `null` to indicate that the new props do not require any state updates.

77. What is the purpose of getSnapshotBeforeUpdate() lifecycle method?

The `getSnapshotBeforeUpdate()` method is a lifecycle method in React that is invoked right before the changes from the virtual DOM are to be reflected in the actual DOM. It allows the component to capture some information from the DOM before it is potentially changed, such as scroll position, and use that information during the `componentDidUpdate()` lifecycle method.

78. Do Hooks replace render props and higher order components?

Yes, Hooks in React can replace render props and higher-order components (HOCs) in many cases. Hooks provide a more composable and flexible way to reuse logic across components without the need for nested component hierarchies or additional wrapper components.

79. What is the recommended way for naming components?

The recommended way for naming components in React is to use descriptive and meaningful names that accurately reflect their purpose or functionality. Component names should be written in PascalCase and start with a capital letter, such as `MyComponent` or `Header`.

80. What is the recommended ordering of methods in component class?

The recommended ordering of methods in a component class in React is as follows:

1. Constructor
2. Static methods (if any)
3. Lifecycle methods (in mounting, updating, and unmounting order)
4. Event handlers

5. Other methods

81. What is a switching component?

A switching component in React refers to a component that conditionally renders different child components based on some condition or state. This pattern is often used for implementing dynamic UIs where different components need to be displayed based on user interactions or application state.

82. Why do we need to pass a function to setState()?

We need to pass a function to `setState()` in React because state updates may be asynchronous. By passing a function, we ensure that we are always working with the latest state value, preventing race conditions or conflicts between concurrent state updates.

83. What is strict mode in React?

Strict mode is a feature in React that helps identify and prevent common issues in your code. It enables additional checks and warnings for potential problems such as deprecated lifecycle methods, unsafe actions, and legacy context API usage. Strict mode is primarily used for debugging and improving code quality.

84. What are React Mixins?

React Mixins are a way to share reusable code between components in React. They allow you to encapsulate logic or behavior and apply it to multiple components without the need for inheritance. However, mixins are considered an advanced feature and have some drawbacks, such as potential conflicts and issues with component composition.

85. Why is isMounted() an anti-pattern and what is the proper solution?

`isMounted()` is considered an anti-pattern in React because it can lead to race conditions and unreliable behavior. It's unreliable because React does not guarantee synchronous updates to component state or lifecycle methods. Instead of `isMounted()`, it's recommended to use instance properties or state flags to track component unmounting and handle asynchronous actions safely.

86. What are the Pointer Events supported in React?

React supports standard pointer events such as `onPointerDown`, `onPointerMove`, `onPointerUp`, `onPointerEnter`, and `onPointerLeave`. These events are similar to mouse events but also support touch and pen input, making them more versatile for handling user interactions in modern web applications.

87. Why should component names start with a capital letter?

Component names in React should start with a capital letter to distinguish them from regular HTML elements and to comply with the naming convention for custom components. This helps React differentiate between built-in elements like `div` or `span` and custom components like `MyComponent`, making the code more readable and consistent.

88. Are custom DOM attributes supported in React v16?

Yes, custom DOM attributes are supported in React v16. You can pass custom attributes to React components as props, and they will be rendered as regular HTML attributes in the DOM. However, it's important to note that custom attributes prefixed with `data-` are recommended to comply with HTML5 specifications and avoid conflicts with future HTML attributes.

89. What is the difference between constructor and getInitialState?

In React, `constructor` is a standard JavaScript method used for initializing class instances, including React components. It's used to set up initial state, bind event handlers, or perform other setup tasks. On the other hand, `getInitialState` is a legacy method used in React class components prior to ES6 class syntax. It's used specifically for initializing component state and is invoked once when the component is created.

90. Can you force a component to re-render without calling setState?

Yes, you can force a component to re-render without calling `setState()` by using the `forceUpdate()` method provided by React. However, it's generally not recommended to use `forceUpdate()` because it bypasses React's state management and can lead to unpredictable behavior or performance issues. It should only be used as a last resort when absolutely necessary.

91. What is the difference between super() and super(props) in React using ES6 classes?

In React using ES6 classes, both `super()` and `super(props)` are used to call the constructor of the parent class. However, `super(props)` also passes props to the parent constructor, allowing you to access `this.props` in the constructor of the subclass. If you don't need to access props in the constructor, you can simply use `super()` without passing props.

92. How to loop inside JSX?

You can loop inside JSX using JavaScript's `map()` function to iterate over an array and return an array of JSX elements. For example:

```jsx
<ul>
  {items.map(item => <li key={item.id}>{item.name}</li>)}
```

```
</ul>
```

93. How do you access props in attribute quotes?

You can access props inside attribute quotes in JSX using curly braces `{}` for interpolation. For example:

```jsx
<input type="text" placeholder={`Enter ${this.props.inputType}`} />
```

94. What is React PropType array with shape?

`PropTypes.arrayOf()` with `PropTypes.shape()` is used to specify an array of objects with specific shapes in React PropTypes. It allows you to define the structure of each object in the array using shape validation. For example:

```javascript
PropTypes.arrayOf(PropTypes.shape({
  name: PropTypes.string,
  age: PropTypes.number
}))
```

95. How to conditionally apply class attributes?

You can conditionally apply class attributes in JSX using a ternary operator or logical AND operator. For example:

```jsx
<div className={isActive ? 'active' : 'inactive'}></div>
```

96. What is the difference between React and ReactDOM?

React is a JavaScript library for building user interfaces, while ReactDOM is a package in React used for interacting with the DOM. React provides the core functionality for creating components, managing state, and handling updates, while ReactDOM handles rendering React components to the DOM and updating the DOM in response to changes.

97. Why is ReactDOM separated from React?

Separating ReactDOM from React allows React to be more versatile and platform-agnostic. React can be used for building not only web applications but also native mobile apps (React Native), virtual reality experiences (React 360), and other platforms. By separating ReactDOM, React can be adapted to different environments without being tightly coupled to the DOM.

98. How to use the React label element?

In React, the `label` element is used to associate text with form elements like `input`, `textarea`, or `select`. You can use the `htmlFor` attribute to specify the `id` of the associated form element. For example:

```jsx
<label htmlFor="inputField">Enter your name:</label>
<input type="text

" id="inputField" />
```

99. How to combine multiple inline style objects?

You can combine multiple inline style objects in JSX using the spread operator (`...`) to merge them into a single object. For example:

```jsx
const style1 = { color: 'red' };
const style2 = { fontSize: '16px' };

<div style={{ ...style1, ...style2 }}>Combined styles</div>
```

100. How to re-render the view when the browser is resized?

To re-render the view when the browser is resized in React, you can listen for the `resize` event using `window.addEventListener()` and update the component's state accordingly. You can then conditionally render components or apply styles based on the updated state. Alternatively, you can use libraries like `react-resize-detector` for handling resize events more efficiently.

101. What is the difference between setState and replaceState methods?

- `setState`: Updates the state of a component by merging the new state object with the existing state.

- `replaceState`: Replaces the entire state of a component with a new state object. It completely discards the existing state and replaces it with the new state.

102. How to listen to state changes?

You can listen to state changes in React by using lifecycle methods like `componentDidUpdate()` or by subscribing to state changes using libraries like Redux. Additionally, you can use React's built-in context API or implement custom event listeners within your components to listen for state changes.

103. What is the recommended approach of removing an array element in react state?

The recommended approach for removing an array element from React state is to use the `filter()` method to create a new array without the element to be removed. This ensures immutability and avoids directly mutating the original state. For example:

```javascript
this.setState(prevState => ({
  array: prevState.array.filter(item => item !== itemToRemove)
}));
```

104. Is it possible to use React without rendering HTML?

Yes, it's possible to use React without rendering HTML. React can be used for rendering components to different platforms and environments, such as native mobile apps (React Native), virtual reality experiences (React 360), or server-side rendering with Node.js. In these cases, React components may render to different formats or representations other than HTML, such as native UI components or VR scenes.

105. How to pretty print JSON with React?

You can pretty print JSON with React by using the `JSON.stringify()` method with indentation. For example:

```javascript
const data = { name: 'John', age: 30 };
const prettyJSON = JSON.stringify(data, null, 2);
console.log(prettyJSON);
```

106. Why you can't update props in React?

In React, props are immutable and cannot be directly modified by the component receiving them. This is because props are passed down from parent components and are intended to be read-only. If a component needs to update its behavior based on changes in props, it should manage its own state and update accordingly.

107. How to focus an input element on page load?

You can focus an input element on page load in React by using the `ref` attribute to create a reference to the input element and then calling the `focus()` method on that reference. For example:

```jsx
class MyComponent extends React.Component {
  constructor(props) {
    super(props);
    this.inputRef = React.createRef();
  }

  componentDidMount() {
    this.inputRef.current.focus();
  }

  render() {
    return <input ref={this.inputRef} />;
  }
}
```

108. What are the possible ways of updating objects in state?

There are several ways to update objects in state in React:

- Using the spread operator (`...`) to create a shallow copy of the object and then updating the properties.

- Using `Object.assign()` to merge the existing object with a new object containing updated properties.

- Using immutable data structures like Immutable.js to ensure immutability when updating state.

110. How can we find the version of React at runtime in the browser?

You can find the version of React at runtime in the browser by accessing the `React.version` property. This property contains the version number of the React library currently loaded in the browser. For example:

```javascript
console.log(React.version);
```

111. What are the approaches to include polyfills in your create-react-app?

There are several approaches to include polyfills in a Create React App:

- Manually import polyfills as needed using a polyfill service like polyfill.io.
- Install and configure polyfills using a package manager like `core-js`, `babel-preset-env`, or `@babel/preset-env`.
- Use the `react-app-polyfill` package provided by Create React App, which includes commonly used polyfills for older browsers.

112. How to use https instead of http in create-react-app?

To use HTTPS instead of HTTP in Create React App, you can set up HTTPS in your development environment by running the following command:

```
HTTPS=true npm start
```

This will start the development server with HTTPS enabled, allowing you to access your React application over a secure connection.

113. How to avoid using relative path imports in create-react-app?

To avoid using relative path imports in Create React App, you can set up module resolution using absolute paths in your project's `jsconfig.json` or `tsconfig.json` file. This allows you to import modules using absolute paths relative to the project root, making imports more consistent and easier to maintain.

114. How to add Google Analytics for react-router?

To add Google Analytics for React Router, you can use the `react-ga` package, which provides a simple way to integrate Google Analytics with React applications. You can track page views and user interactions by calling the appropriate methods from `react-ga` in your components or route handlers.

115. How to update a component every second?

You can update a component every second in React by using `setInterval()` to trigger a state update at regular intervals. For example:

```javascript
componentDidMount() {
  this.interval = setInterval(() => {
    // Update component state
    this.setState({ currentTime: new Date() });
  }, 1000);
}

componentWillUnmount() {
  clearInterval(this.interval);
}
```

116. How do you apply vendor prefixes to inline styles in React?

You can apply vendor prefixes to inline styles in React using libraries like `inline-style-prefixer` or `autoprefixer`. These libraries automatically add necessary vendor prefixes to CSS properties based on browser compatibility data. Alternatively, you can manually add vendor prefixes using JavaScript or CSS preprocessors.

117. How to import and export components using react and ES6?

You can import and export components using ES6 modules in React by using the `import` and `export` keywords. For example:

```javascript
// Exporting a component
export class MyComponent extends React.Component {
  // Component logic here
}


// Importing a component
import { MyComponent } from './MyComponent';
```

118. What are the exceptions on React component naming?

In React, component names must start with a capital letter and follow PascalCase naming convention. However, there are exceptions for certain types of components:

   - Native HTML elements like `div`, `span`, or `input` should be lowercase.

   - Higher-order components (HOCs) or functional components can use descriptive names but should still follow the capitalization convention.

119. Why is a component constructor called only once?

The component constructor is called only once during the component's lifecycle when it is initially created. This is because the constructor is responsible for initializing the component's state and setting up initial configurations. Subsequent renders of the component reuse the existing instance, so the constructor is not called again.

120. How to define constants in React?

You can define constants in React by declaring them at the top of your component file or in a separate constants file using the `const` keyword. Constants should be defined outside of the component class or function to ensure they are not redefined

 on each render. For example:

```javascript
const MAX_COUNT = 10;


class MyComponent extends React.Component {
  // Component logic here
}
```

These constants can then be used throughout your component or application as needed.

121. How to programmatically trigger click event in React?

You can programmatically trigger a click event in React by using the `click()` method on a ref to the DOM element you want to click. First, create a ref using `React.createRef()`, then attach it to the element you want to click. Finally, call the `click()` method on the ref. For example:

```javascript
class MyComponent extends React.Component {
  constructor(props) {
    super(props);
    this.myRef = React.createRef();
  }

  handleClick = () => {
    this.myRef.current.click();
  }

  render() {
    return <button ref={this.myRef}>Click me</button>;
  }
}
```

Calling `handleClick()` will programmatically trigger a click event on the button.

122. Is it possible to use async/await in plain React?

Yes, it is possible to use `async/await` in plain React. You can use `async/await` syntax inside lifecycle methods, event handlers, or any other asynchronous function in your React components. However, you should be cautious about using `async` functions in lifecycle methods, as they can affect component rendering and performance.

123. What are the common folder structures for React?

Common folder structures for React projects often include directories like `src`, `components`, `containers`, `assets`, `utils`, `styles`, and `tests`. However, the specific folder structure can vary depending on the project's size, complexity, and organizational preferences.

124. What are the popular packages for animation?

Some popular packages for animation in React include:

- `react-spring`: A spring-physics based animation library for React.
- `framer-motion`: A library for creating fluid animations in React.
- `react-transition-group`: A library for managing animations during transitions in React components.

125. What is the benefit of styles modules?

Style modules, also known as CSS modules, provide scoped styling for React components by generating unique class names for each component. This helps prevent CSS class name collisions and allows for more modular and maintainable styling. Additionally, style modules support features like local scope, composition, and automatic vendor prefixing.

126. What are the popular React-specific linters?

Some popular linters for React-specific code include:

- `eslint-plugin-react`: A plugin for ESLint that provides React-specific linting rules.
- `eslint-plugin-react-hooks`: A plugin for ESLint that provides rules for React Hooks.
- `eslint-config-airbnb`: A popular ESLint configuration that includes React-specific rules and conventions.

127. How to make AJAX call and In which component lifecycle methods should I make an AJAX call?

You can make AJAX calls in React using libraries like Axios, Fetch API, or native XMLHttpRequest. The recommended lifecycle method for making AJAX calls in React is `componentDidMount()`, as it ensures that the component has been mounted and is ready to fetch data. However, you can also make AJAX calls in other lifecycle methods like `componentDidUpdate()` if the data needs to be updated in response to props or state changes.

128. What are render props?

Render props is a pattern in React where a component's `render` method receives a function as its children prop. This function, often called the "render prop," allows the parent component to control what is rendered by the child component. Render props are commonly used for sharing code or behavior between components in a flexible and composable way.