Profiling

Profiling in React is crucial for understanding how your components render and ensuring that your application runs efficiently. React provides several tools and techniques to help with profiling.

React.lazy

React.lazy is a feature in React that allows you to load components lazily through code splitting. This can improve the performance of your application by reducing the initial load time, as it only loads the components when they are needed. This is particularly useful for large applications where you want to load different parts of your application on demand.

Hooks

useState

The React `useState` Hook allows us to track state in a function component.

State generally refers to data or properties that need to be tracking in an application.

To use the `useState` Hook, we first need to `import` it into our component.

`useState` accepts an initial state and returns two values:

- The current state.
- A function that updates the state.

When state is updated, the entire state gets overwritten.

UseEffect

The `useEffect` Hook allows you to perform side effects in your components.

Some examples of side effects are: fetching data, directly updating the DOM, and timers.

useContext

React Context is a way to manage state globally.

It can be used together with the `useState` Hook to share state between deeply nested components more easily than with `useState` alone.

useRef

The `useRef` Hook allows you to persist values between renders.

It can be used to store a mutable value that does not cause a re-render when updated.

It can be used to access a DOM element directly.

`useRef()` only returns one item. It returns an Object called `current`.

useReducer

The `useReducer` Hook is similar to the `useState` Hook.

It allows for custom state logic.

If you find yourself keeping track of multiple pieces of state that rely on complex logic, `useReducer` may be useful.

useCallback

The React `useCallback` Hook returns a memoized callback function.

Think of memoization as caching a value so that it does not need to be recalculated.

This allows us to isolate resource intensive functions so that they will not automatically run on every render.

The `useCallback` Hook only runs when one of its dependencies update.

This can improve performance.

The `useCallback` and `useMemo` Hooks are similar. The main difference is that `useMemo` returns a memoized *value* and `useCallback` returns a memoized *function*.

useMemo

The React `useMemo` Hook returns a memoized value.

Think of memoization as caching a value so that it does not need to be recalculated.

The `useMemo` Hook only runs when one of its dependencies update.

This can improve performance.

The `useMemo` and `useCallback` Hooks are similar. The main difference is that `useMemo` returns a memoized value and `useCallback` returns a memoized function.

Custom Hooks

A custom hook in React is a JavaScript function that starts with `use` and lets you use React state and other features in functional components. Custom hooks are a way to extract component logic into reusable functions.

componentDidUpdate, componentDidMount and componentWillUnmount use in hooks

In React functional components with hooks, you can achieve similar functionality to componentDidMount, componentDidUpdate, and componentWillUnmount lifecycle methods using the useEffect hook. Here's how you can use useEffect to replicate these lifecycle behaviors:

useEffect Hook Equivalent for componentDidMount

In class components, componentDidMount is invoked immediately after a component is mounted (inserted into the DOM tree). You can achieve the same effect in functional components using useEffect with an empty dependency array ([]). This ensures that the effect runs only once, after the initial render:

useEffect Hook Equivalent for componentDidUpdate
In class components, componentDidUpdate is invoked immediately after an update occurs (props or state changes). You can replicate this behavior using useEffect with a dependency array that includes the variables [var/value] you want to monitor for changes:

useEffect for Mounting == (componentDidMount equivalent)
useEffect for Updating == (componentDidUpdate equivalent)
Cleanup == (componentWillUnmount equivalent)

forwardRef

In React, forwardRef is a function that lets you pass a ref through a component to one of its children. This can be useful when you need to access a DOM element or a child component's instance from a parent component. Here's a basic example to illustrate how forwardRef works:

prop drilling

Prop drilling is a common term in React development that refers to the process of passing data from a parent component to a deeply nested child component through multiple layers of intermediate components. This often results in a cumbersome and less maintainable codebase, especially as the depth of the component tree increases.

1. callback hooks

https://www.w3schools.com/react/react_usecallback.asp
https://react.dev/reference/react/useCallback
https://www.geeksforgeeks.org/react-js-usecallback-hook/

The React useCallback Hook returns a memoized callback function.
Think of memoization as caching a value so that it does not need to be recalculated.
This allows us to isolate resource intensive functions so that they will not automatically run on every render.
The useCallback Hook only runs when one of its dependencies update.
This can improve performance.

useCallback is a React Hook that lets you cache a function definition between re-renders.

Usage
Skipping re-rendering of components

Updating state from a memoized callback
Preventing an Effect from firing too often
Optimizing a custom Hook

Troubleshooting
Every time my component renders, useCallback returns a different function
I need to call useCallback for each list item in a loop, but it's not allowed

Call useCallback at the top level of your component to cache a function definition between re-renders:

React useCallback hook returns a memoized function to reduce unnecessary callbacks. This useCallback hook is used when you have a component in which the child is rerendering again and again without need.

2. custom hooks

https://www.w3schools.com/react/react_customhooks.asp
https://legacy.reactjs.org/docs/hooks-custom.html
https://www.geeksforgeeks.org/reactjs-custom-hooks/

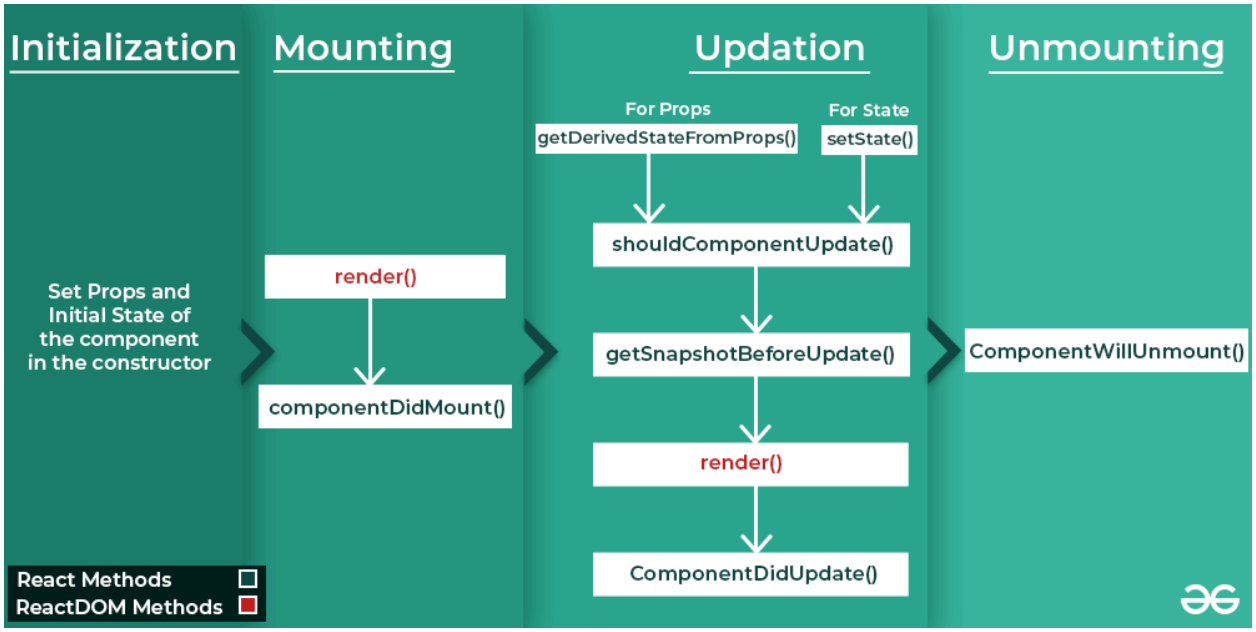
React Custom Hooks are JavaScript functions starting with 'use' containing reusable logic.

We know that hooks like useState, and useEffect are reusable components. Sometimes we make components that we have to reuse again and again in the application. In this case, we can convert the component to hooks by extracting logic from it.

The main reason why you should be using Custom hooks is to maintain the concept of DRY(Don't Repeat Yourself) in your React apps.

3. React LIfecycle

https://www.w3schools.com/react/react_lifecycle.asp
https://www.geeksforgeeks.org/reactjs-lifecycle-components/
https://www.tutorialspoint.com/reactjs/reactjs_component_life_cycle.htm
https://www.geeksforgeeks.org/explain-lifecycle-methods-of-react-components/



Each React component has three distinct stages.

Mounting − Mounting represents the rendering of the React component in the given DOM node.

Updating − Updating represents the re-rendering of the React component in the given DOM node during state changes / updates.

Unmounting − Unmounting represents the removal of the React component.

4. child to parent component call

https://medium.com/@s.salman0193/calling-parent-components-function-from-child-component-in-react-js-e2d550ccd66b#:~:text=Method%201%3A%20Callback%20Functions,call%20from%20the%20child%20component.
https://www.geeksforgeeks.org/how-to-pass-data-from-child-component-to-its-parent-in-reactjs/
https://www.geeksforgeeks.org/how-to-call-parent-componentss-function-from-child-component-in-react/

Following are the steps to pass data from the child component to the parent component:

In the parent component, create a callback function. This callback function will retrieve the data from the child component.

Pass the callback function to the child as a prop from the parent component.

The child component calls the parent callback function using props and passes the data to the parent component.

Method 1: Callback Functions
Method 2: Using refs

5. useState

https://www.geeksforgeeks.org/reactjs-usestate-hook/
https://www.w3schools.com/react/react_usestate.asp
https://www.geeksforgeeks.org/what-is-usestate-in-react/

The React useState Hook allows us to track state in a function component.

State generally refers to data or properties that need to be tracking in an application.

6. class base and function based

https://www.geeksforgeeks.org/differences-between-functional-components-and-class-components/
https://blog.geekster.in/functional-components-vs-class-components/
https://www.reddit.com/r/reactjs/comments/zqo6th/functional_vs_class_components/

Functional Components	Class Components
A functional component is just a plain JavaScript pure function that accepts props as an argument and returns a React element(JSX).	A class component requires you to extend from React. Component and create a render function that returns a React element.
There is no render method used in functional components.	It must have the render() method returning JSX (which is syntactically similar to HTML)
Functional components run from top to bottom and once the function is returned it can't be kept alive.	The class component is instantiated and different life cycle method is kept alive and is run and invoked depending on the phase of the class component.
Also known as Stateless components as they simply accept data and display them in some form, they are mainly responsible for rendering UI.	Also known as Stateful components because they implement logic and state.
React lifecycle methods (for example, componentDidMount) cannot be used in functional components.	React lifecycle methods can be used inside class components (for example, componentDidMount).
Hooks can be easily used in functional components to make them Stateful. Example: `const [name,SetName]= React.useState(' ')`	It requires different syntax inside a class component to implement hooks. Example: `constructor(props) { super(props); this.state = {name: ' '} }`

	Functional Components	Class Components
Syntax	Functional components are written as a JavaScript function.	Class components are written as a JavaScript class.
State and Lifecycle Methods	Functional components do not have a state or lifecycle methods.	Class components have a state and can implement lifecycle methods like componentDidMount and componentDidUpdate.
Performance	Faster as they do not have state and lifecycle, react needs to do less work to render these components.	Slower as they have state and lifecycle, react needs to do comparatively more work to render these components.
Code Length	Functional components tend to be shorter and more concise	Class components require the boilerplate code, such as a constructor method and the use of "this" to access props and state.
Usage of "this"	Functional components do not use "this" at all, which makes them easier to understand for beginners.	Class components use the "this" keyword is used to refer to the current instance of the component which can be confusing for new developers.

7. props

https://www.w3schools.com/react/react_props.asp
https://www.geeksforgeeks.org/reactjs-props-set-1/
https://legacy.reactjs.org/docs/components-and-props.html

React allows us to pass information to a Component using something called props (which stands for properties). Props are objects which can be used inside a component.

Props are arguments passed into React components.
props stands for properties.

React Props are like function arguments in JavaScript and attributes in HTML.

To send props into a component, use the same syntax as HTML attributes:

props are immutable, meaning once they are set by the parent component, they cannot be changed or modified by the child component.

8. router

https://www.w3schools.com/react/react_router.asp
https://www.tutorialspoint.com/reactjs/reactjs_routing.htm

Routing is a process of binding a web URL to a specific resource in the web application. In React, it is binding an URL to a component. React does not support routing natively as it is basically an user interface library. React community provides many third party component to handle routing in the React application. Let us learn React Router, a top choice routing library for React application.

React router provides four components to manage navigation in React application.

Router − Router is th top level component. It encloses the entire application.

Link − Similar to anchor tag in html. It sets the target url along with reference text.

<Link to="/">Home</Link>
Here, to attribute is used to set the target url.

Route − Maps the target url to the component.

9. hook

https://www.w3schools.com/react/react_hooks.asp
https://www.geeksforgeeks.org/reactjs-hooks/
Hooks are reusable functions that provide access to state in React Applications. Hooks were introduced in the 16.8 version of React. Hooks give access to states for functional components while creating a React application. It allows you to use state and other React features without writing a class.

10. contextApi

https://www.w3schools.com/react/react_usecontext.asp
https://www.freecodecamp.org/news/context-api-in-react/
https://www.geeksforgeeks.org/explain-new-context-api-in-react/
https://blog.logrocket.com/react-context-api-deep-dive-examples/

Context API is used to pass global variables anywhere in the code. It helps when there is a need for sharing state between a lot of nested components. It is light in weight and easier to use, to create a context just need to call React.createContext().

React Context to avoid prop drilling

React Context is a way to manage state globally.

11. useCallback

https://www.w3schools.com/react/react_usecallback.asp
https://www.geeksforgeeks.org/react-js-usecallback-hook/

React useCallback hook returns a memoized function to reduce unnecessary callbacks. This useCallback hook is used when you have a component in which the child is rerendering again and again without need.

The React useCallback Hook returns a memoized callback function.

Think of memoization as caching a value so that it does not need to be recalculated.

This allows us to isolate resource intensive functions so that they will not automatically run on every render.

The useCallback Hook only runs when one of its dependencies update.

This can improve performance.

12. jsx

https://www.w3schools.com/react/react_jsx.asp
https://www.geeksforgeeks.org/reactjs-jsx-introduction/
https://www.freecodecamp.org/news/jsx-in-react-introduction/
JSX is a JavaScript Extension Syntax used in React to easily write HTML and JavaScript together.

JSX stands for JavaScript XML. JSX is basically a syntax extension of JavaScript.

React JSX helps us to write HTML in JavaScript and forms the basis of React Development. Using JSX is not compulsory but it is highly recommended for programming in React as it makes the development process easier as the code becomes easy to write and read.

JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement() and/or appendChild() methods.

JSX converts HTML tags into react elements.

13. render

https://www.w3schools.com/react/react_render.asp
https://www.geeksforgeeks.org/explain-the-purpose-of-render-in-reactjs/

It is used to display the component on the UI returned as HTML or JSX components. The ReactDOM.render() function takes two arguments, HTML code and an HTML element.

14. onclick button show increment and decrement

https://www.geeksforgeeks.org/how-to-make-incremental-and-decremental-counter-using-html-css-and-javascript/

15. react features

https://www.geeksforgeeks.org/what-are-the-features-of-reactjs/
https://www.tutorialspoint.com/reactjs/reactjs_features.htm
Virtual DOM
Components
JSX
One way data binding
Scalable
Flexible
Modular

16. Can web browsers read JSX directly?

https://www.geeksforgeeks.org/why-cant-browsers-read-jsx/
https://dev.to/cloud_developr/why-can-t-browsers-read-jsx-5boc
https://iq.js.org/questions/react/do-browsers-understand-jsx-code#:~:text=No%2C%20browsers%20can't%20understand,transpiler%20right%20now%20is%20Babel.

No, browsers can't understand JSX code. You need a transpiler to convert your JSX to regular Javascript that browsers can understand. The most widely used transpiler right now is Babel.

Browsers can only read JavaScript objects but JSX in not a regular JavaScript object. Thus to enable a browser to read JSX, first, we need to transform JSX file into a JavaScript object using JSX transformers like Babel and then pass it to the browser.

17. virtual DOM

https://www.geeksforgeeks.org/reactjs-virtual-dom/
https://www.geeksforgeeks.org/how-to-understand-by-virtual-dom/
https://legacy.reactjs.org/docs/faq-internals.html

The Virtual DOM is an in-memory representation of the DOM. DOM refers to the Document Object Model that represents the content of XML or HTML documents as a tree structure so that the programs can be read, accessed and changed in the document structure, style, and content.

The name itself says that it is a virtually created DOM. Virtual DOM is exactly like DOM and it has all the properties that DOM has. But the main difference is Whenever a code runs JavaScript Framework updates the whole DOM at once which gives a slow performance. whereas virtual DOM updates only the modified part of the DOM.

When you run a code, the web page is divided into different modules. So, virtual DOM compares it with DOM and checks if there is any difference. If it finds a difference then DOM updates only the modified part and the other part remains the same.

Virtual DOM Key Concepts :
Virtual DOM is the virtual representation of Real DOM
React update the state changes in Virtual DOM first and then it syncs with Real DOM
Virtual DOM is just like a blueprint of a machine, can do changes in the blueprint but those changes will not directly apply to the machine.
Virtual DOM is a programming concept where a virtual representation of a UI is kept in memory synced with “Real DOM ” by a library such as ReactDOM and this process is called reconciliation
Virtual DOM makes the performance faster, not because the processing itself is done in less time. The reason is the amount of changed information – rather than wasting time on updating the entire page, you can dissect it into small elements and interactions

18. state and props

https://www.geeksforgeeks.org/what-are-the-differences-between-props-and-state/
https://www.javatpoint.com/react-state-vs-props
https://www.geeksforgeeks.org/reactjs-state-vs-props/

PROPS	STATE
The Data is passed from one component to another.	The Data is passed within the component only.
It is Immutable (cannot be modified).	It is Mutable (can be modified).
Props can be used with state and functional components.	The state can be used only with the state components/class component (Before 16.0).
Props are read-only.	The state is both read and write.

SN	Props	State
1.	Props are read-only.	State changes can be asynchronous.
2.	Props are immutable.	State is mutable.
3.	Props allow you to pass data from one component to other components as an argument.	State holds information about the components.
4.	Props can be accessed by the child component.	State cannot be accessed by child components.
5.	Props are used to communicate between components.	States can be used for rendering dynamic changes with the component.
6.	Stateless component can have Props.	Stateless components cannot have State.
7.	Props make components reusable.	State cannot make components reusable.
8.	Props are external and controlled by whatever renders the component.	The State is internal and controlled by the React Component itself.

19. higher-order component

https://www.geeksforgeeks.org/react-js-higher-order-components/
https://legacy.reactjs.org/docs/higher-order-components.html
https://www.freecodecamp.org/news/higher-order-components-in-react/
https://www.geeksforgeeks.org/how-to-use-higher-order-components-in-react/

Higher-order components or HOC is the advanced method of reusing the component functionality logic. It simply takes the original component and returns the enhanced component.

Reasons to use Higher-Order Components:

Code Reusability: Higher components encapsulate to reuse the common logic here. This we do across many components to reduce code duplication.

Abstraction of Logic: Higher Order Component often removes the repetitive tasks. Such as data fetching or authentication. This makes our code to have more code readability.

Composability: Several behaviors or features can be combined into single components using several HOCs hence providing granular control over component behavior and allowing for the creation of sophisticated UIs.

20. lifecycle methods of components

https://www.w3schools.com/react/react_lifecycle.asp
https://www.geeksforgeeks.org/reactjs-lifecycle-components/
https://www.geeksforgeeks.org/explain-lifecycle-methods-of-react-components/

21. What is React and how does it differ from other JavaScript frameworks/libraries?

https://codedamn.com/news/reactjs/how-is-react-different-from-other-javascript-frameworks-like-angular-or-vue
https://javascript.plainenglish.io/how-is-reactjs-different-from-other-javascript-libraries-494b56459344
https://www.geeksforgeeks.org/why-is-react-considered-a-library-and-not-a-framework/

React was developed and is maintained by Facebook, released in 2013. It was born out of the company's need to manage increasing amounts of data in real-time and ensure a seamless user experience. React quickly became popular due to its component-based architecture and efficient rendering with the virtual DOM. Facebook's own platforms, like Instagram, are built using React.

22. What are the benefits of using React?

https://www.geeksforgeeks.org/what-are-the-advantages-of-react-js/
https://www.javatpoint.com/pros-and-cons-of-react
https://www.geeksforgeeks.org/what-are-the-advantages-of-using-redux-with-reactjs/
https://www.devstringx.com/pros-and-cons-of-using-react

The advantages of React JS are :
It is composable.
It is declarative.
Write once, and learn anywhere.
It is simple.
SEO friendly.
Fast, efficient, and easy to learn.
It guarantees stable code.
It is backed by a strong community.

23. How do you handle events in React?

https://www.w3schools.com/react/react_events.asp
https://legacy.reactjs.org/docs/handling-events.html
https://www.freecodecamp.org/news/how-to-handle-events-in-react-19/

Handling events in React is similar to handling events in plain HTML or JavaScript, but with some differences due to the use of JSX and the React component model. Here's a comprehensive guide on how to handle events in React:

1. Basic Event Handling

2. Passing Arguments to Event Handlers

3. Synthetic Events

4. Using State with Event Handlers

5. Handling Events in Class Components

Using arrow functions (no binding needed):

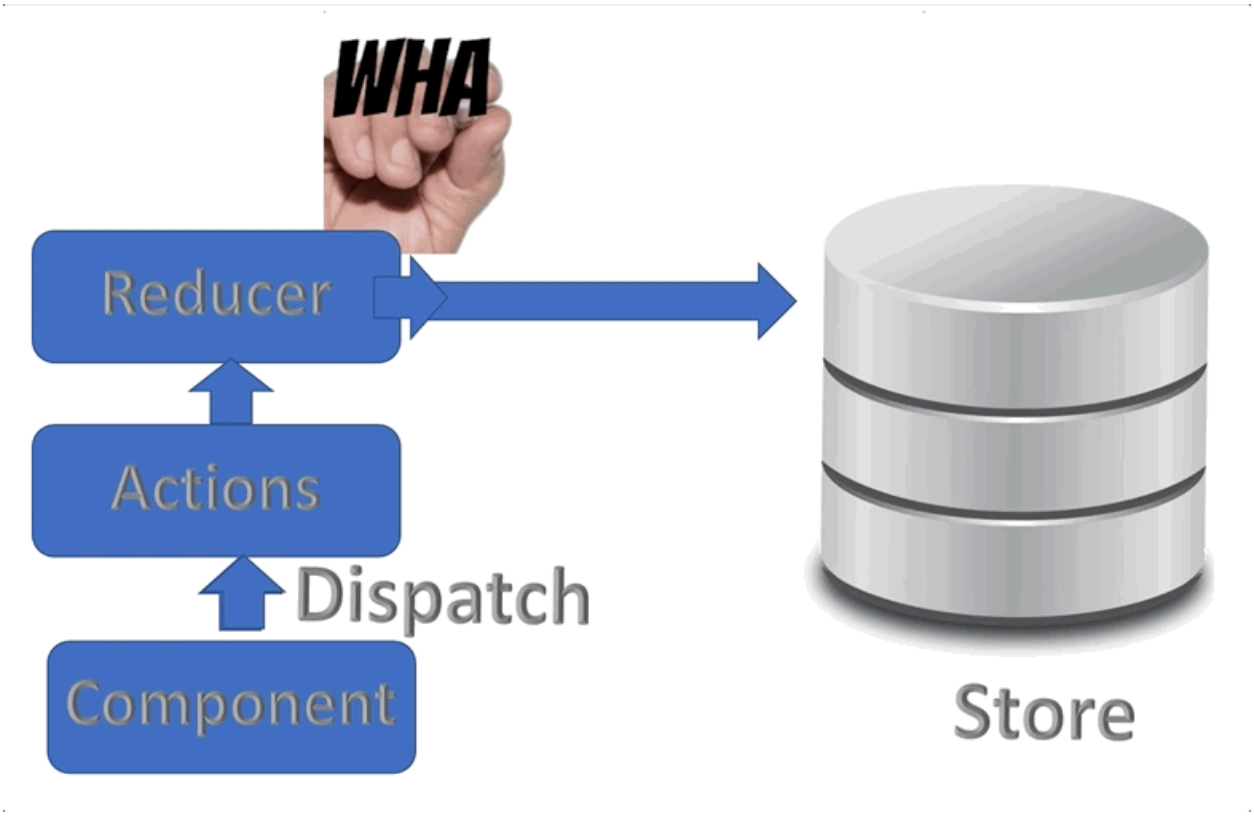
6. Event Pooling

React supports many different types of events, including:

Mouse events (onClick, onDoubleClick, onMouseEnter, onMouseLeave, etc.)
Form events (onChange, onSubmit, onFocus, onBlur, etc.)
Keyboard events (onKeyDown, onKeyPress, onKeyUp, etc.)
Focus events (onFocus, onBlur)
Touch events (onTouchStart, onTouchMove, onTouchEnd, etc.)

24. What is Redux and how is it used in React?

https://www.geeksforgeeks.org/what-are-the-advantages-of-using-redux-with-reactjs/
https://www.tutorialspoint.com/reactjs/reactjs_redux.htm
https://www.geeksforgeeks.org/react-redux-tutorial/
https://medium.com/swlh/what-is-redux-b16b42b33820



it provides a predictable state container by connecting React components to a centralized store, simplifying data flow and enabling efficient management of application state across components.

Advantages:
Centralized state management system i.e. Store
Performance Optimizations
Pure reducer functions
Storing long-term data
Great supportive community

React redux maintains the state of the application in a single place called Redux store. React component can get the latest state from the store as well as change the state at any time. Redux provides a simple process to get and set the current state of the application and involves below concepts.

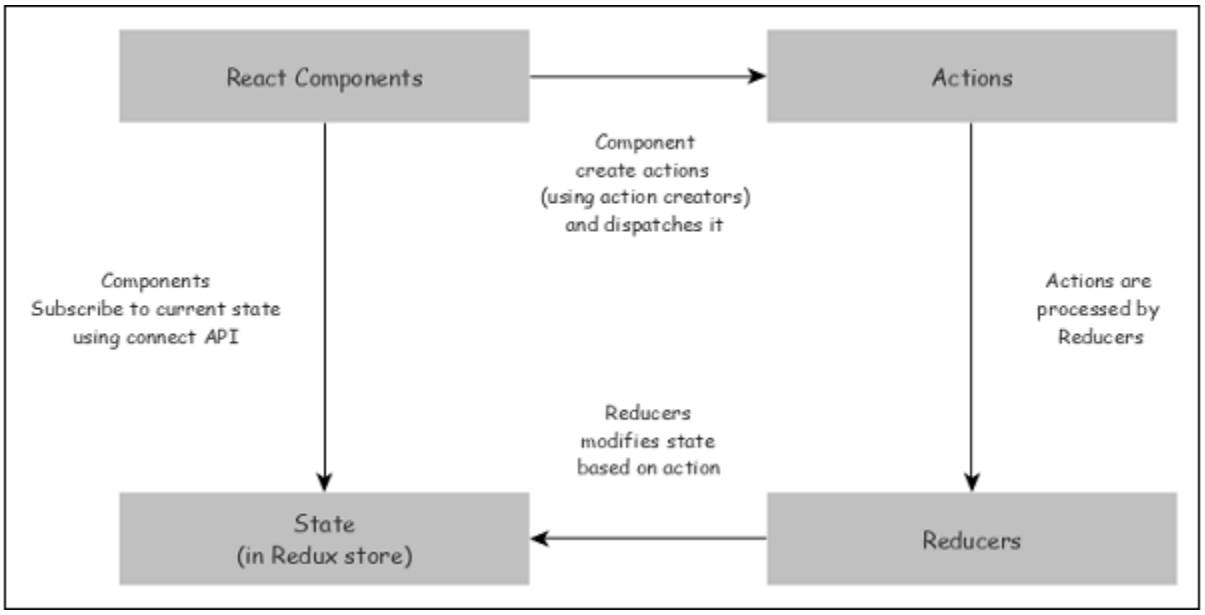
Store − The central place to store the state of the application.

Action- for adding an item in the store contains ADD_ITEM as type and an object with item's details as payload.

Reducers − Reducers are pure functions used to create a new state based on the existing state and the current action.

Action creators − Action creator creates an action with proper action type and data necessary for the action and returns the action.

Component − Component can connect to the store to get the current state and dispatch action to the store so that the store executes the action and updates it's current state.



React component subscribes to the store and get the latest state during initialization of the application.
To change the state, React component creates necessary action and dispatches the action.
Reducer creates a new state based on the action and returns it. Store updates itself with the new state.
Once the state changes, store sends the updated state to all its subscribed component.

25. controlled and uncontrolled components in React.

https://www.scaler.com/topics/react/controlled-and-uncontrolled-components-in-react/#:~:text=In%20a%20controlled%20component%20react,and%20where%20to%20insert%20it.
https://www.geeksforgeeks.org/controlled-vs-uncontrolled-components-in-reactjs/
https://www.javatpoint.com/react-controlled-vs-uncontrolled-component
https://medium.com/tech-tajawal/controlled-and-uncontrolled-components-in-react-6d5f260b46dd
https://www.geeksforgeeks.org/what-are-controlled-components-in-reactjs/

Controlled Component	Uncontrolled Component
The component is under control of the component's state.	Components are under the control of DOM.
These components are predictable as are controlled by the state of the component.	Are Uncontrolled because during the life cycle methods the data may loss
Internal state is not maintained	Internal state is maintained
It accepts the current value as props	We access the values using refs
Does not maintain its internal state.	Maintains its internal state.
Controlled by the parent component.	Controlled by the DOM itself.
Have better control on the form data and values	Has very limited control over form values and data

Controlled	Uncontrolled
It does not maintain its internal state.	It maintains its internal states.
Here, data is controlled by the parent component.	Here, data is controlled by the DOM itself.
It accepts its current value as a prop.	It uses a ref for their current values.
It allows validation control.	It does not allow validation control.
It has better control over the form elements and data.	It has limited control over the form elements and data.

26. server-side rendering (SSR) and why is it important in React?

https://www.tutorialspoint.com/a-guide-to-server-side-rendering-in-react
https://www.geeksforgeeks.org/what-is-server-side-rendering-in-react/
https://medium.com/simform-engineering/how-to-implement-ssr-server-side-rendering-in-react-18-e49bc43e9531

Server-Side Rendering (SSR) in React is a technique that involves rendering React components on the server side instead of the client side (browser). Traditionally, React applications are rendered on the client side, meaning that the browser downloads the JavaScript bundle,

executes it, and renders the UI. In contrast, with SSR, the server pre-renders the React components into HTML before sending it to the client.

Benefits of Server-Side Rendering:
Improved Initial Page Load Performance: SSR can improve the initial page load performance by sending a fully-rendered HTML page to the client. This can result in faster perceived load times, especially for users with slower internet connections or devices.

Search Engine Optimization (SEO): Search engines often struggle with indexing content rendered on the client side using JavaScript. SSR provides a solution by sending pre-rendered HTML to search engines, making it easier for them to crawl and index the content. This improves the SEO of React applications.

Enhanced User Experience: Users might see content quicker, as the server can send a partially or fully rendered page while the client-side JavaScript is still loading. This can result in a more interactive and engaging user experience.
Support for Browsers with Limited JavaScript: SSR ensures that your application is accessible to users with browsers that have limited or disabled JavaScript support. By delivering pre-rendered HTML, the content is still viewable even if the client cannot execute JavaScript.

27. optimize React performance?

https://www.geeksforgeeks.org/how-to-optimize-the-performance-of-react-app/
https://www.freecodecamp.org/news/react-performance-optimization-techniques/
https://hackernoon.com/best-practices-for-react-performance-optimization
https://www.geeksforgeeks.org/optimizing-performance-in-reactjs/

1. Use React.memo for Component Memoization

2. Use useMemo and useCallback Hooks

3. Split Code with React.lazy and Suspense

4. Optimize Rendering with Keys

5. Avoid Inline Functions and Objects

6. Use the Production Build

7. Profile and Analyze Performance

8. Optimize State Management

10. Optimize CSS and Assets

28. concept of hooks in React.

https://legacy.reactjs.org/docs/hooks-overview.html#:~:text=Hooks%20are%20functions%20that%20let,if%20you'd%20like.)
https://www.w3schools.com/react/react_hooks.asp
https://www.freecodecamp.org/news/react-hooks-fundamentals/
https://www.geeksforgeeks.org/introduction-to-react-hooks/

29. best practices for writing React code?

https://www.freecodecamp.org/news/best-practices-for-react/
https://www.tutorialspoint.com/reactjs/reactjs_best_practices.htm

30. session storage

https://developer.mozilla.org/en-US/docs/Web/API/Window/sessionStorage
https://www.w3schools.com/jsref/prop_win_sessionstorage.asp
https://www.freecodecamp.org/news/web-storage-localstorage-vs-sessionstorage-in-javascript/

31. higher order function

https://www.freecodecamp.org/news/higher-order-functions-what-they-are-and-a-react-example-1d2579faf101/

In React, higher-order functions (HOFs) are functions that take a component (or multiple components) as input and return a new enhanced component. They are commonly used for code reuse, abstraction of logic, and enhancing component behaviors without modifying the original component directly. Here's how higher-order functions are typically used in React:

1. Enhancing Components with HOCs

2. Common Patterns with HOCs
Adding Props and Functionality
HOCs can add props or functionality to components, such as authentication, data fetching, or context handling.

3. Using HOCs vs. Hooks
With the advent of React Hooks, many patterns that used to require HOCs can now be achieved using custom hooks (useXYZ functions). Hooks provide a more direct way to reuse stateful logic between components without introducing an additional layer of nesting like HOCs do.

Summary

Higher-order functions (HOCs) in React are functions that take a component and return a new component with enhanced functionality.
They are useful for adding props, handling conditional rendering, and abstracting common logic across components.
With the advent of React Hooks, many patterns that used to require HOCs can now be achieved using custom hooks (useXYZ functions) for more direct and composable logic reuse.

32. react suspense and lazy

https://legacy.reactjs.org/docs/code-splitting.html
https://www.freecodecamp.org/news/react-suspense/
https://dev.to/tmns/uselazyload-code-splitting-in-react-without-suspense-4e2o

React Suspense and React.lazy are two features introduced in React to help with code splitting and improving loading times of components in React applications.

React.lazy
React.lazy allows you to dynamically import a component only when it is needed, typically for improving the initial loading performance of your application by splitting the bundle into smaller chunks.

React Suspense
React Suspense is a component that allows you to suspend rendering while waiting for some asynchronous operation to complete, such as fetching data or loading components using React.lazy.

Key Points:
React.lazy and Suspense work together to enable dynamic code splitting and improve the performance of React applications by loading components and data only when they are needed.
React.lazy is used to lazy-load components asynchronously.
Suspense is used to handle the loading state of components or data fetching operations and provide a fallback UI until the data or components are ready.

33. webpacks

https://dev.to/mbarzeev/everything-you-need-to-know-about-webpacks-bundle-analyzer-g0l
https://blog.jakoblind.no/webpack-bundle-analyzer/

Webpack is a powerful tool for bundling and optimizing JavaScript applications, including React applications. It simplifies asset management, enables code splitting, and provides a robust ecosystem of plugins and loaders to enhance development and production workflows.
Configuring Webpack for React involves defining entry points, loaders for different file types,

plugins for additional functionality, and setting up a development server for efficient local development.

34. Code Splitting

Code-Splitting is a feature supported by bundlers like Webpack, Rollup and Browserify (via factor-bundle) which can create multiple bundles that can be dynamically loaded at runtime.

Code splitting is a **powerful technique to optimize the performance of React applications**.

Code splitting **uses React.lazy and Suspense tool/library**, which helps you to load a dependency lazily and only load it when needed by the user.

Code splitting is a way to split up your code from a large file into smaller code bundles. These can then be requested on demand or in parallel

Code-splitting your app can help you “lazy-load” just the things that are currently needed by the user, which can dramatically improve the performance of your app. While you haven’t reduced the overall amount of code in your app, you’ve avoided loading code that the user may never need, and reduced the amount of code needed during the initial load.