I am particularly drawn to the innovative culture and forward-thinking approach that your company embodies. Your commitment to pushing boundaries and creating impactful solutions resonates deeply with my own professional ethos. The prospect of being part of such a dynamic team, where creativity is encouraged and innovation is celebrated, fills me with excitement. I am eager to contribute my skills and expertise to help shape the future within your organisation and make meaningful contributions to our collective success.

**HTML**

• layout elements, inline & block, i frames, input tags with types like tel, mailto for calling, mail etc, they can have value attribute set with min/max value, placeholder etc, localStorage/sessionStorage - localStorage.setItem() or getItem() or removeItem() & same for sessionStorage. <iframe name="iframe\_a"> & < a href=" " target="iframe\_a" > This opens the website inside the frame box in the webpage.

**CSS**

• positions, box model, pseudo Elements, grid(grid-template-column & grid-column), flex box(flex wrap).

**JS**

• Synchronous, single threaded by default, let/const/var. JS type conversion(casting) & coercion, spread n rest operators, errors and type, hoisting, closure, **Promises** - A promise is an object that represents the eventual completion or failure of an asynchronous operation & Async/await(Extension of promises), Defer, Event loop in JS, Shallow copy vs deep copy, difference between == & ===, currying, isNaN(Not a Number) , ES5 vs ES6 vs ES7 to ES12, higher order functions, pure function(map, filter, reduce, forEach), Generators(function\* & yield), Debouncing & Throttling, Temporal dead zone

• JS is dynamically typed lang i.e. data conversion during runtime(before compile time). JS has Implicit type coercion (without changing DT & adding Ex. 5+"9"=59) & Explicit type conversion(user can change). Also JS is Synchronous(single thread by nature) in nature by default. To make it asynchronous, promises, async/await are used.

• An immediately invoked function expression (IIFE) is a JavaScript function that runs as soon as it is defined. IIFEs are created by wrapping a function expression in parentheses and then following it with another pair of parentheses.Example: (function(){ // code to be executed immediately})(); , function statement vs function expression, e.preventDefault(),

• When to use call, bind, apply ?

• How does browser reads code and Babel?

• **useStrict** provides benefits like catching common coding errors, preventing the use of undeclared variables, disallowing duplicate parameter names, prohibiting unsafe actions, and improving code performance and security.

• Mutation refers to the act of changing or modifying an object or data structure directly.

A **WebSocket** is a communication protocol that provides full-duplex communication channels over a single, long-lived connection between a client (usually a web browser) and a server. Unlike the traditional HTTP protocol, which follows a request-response model where the client initiates a request and the server responds, WebSockets allow for continuous, bidirectional communication between the client and server.

**REACTJS**

• All Hooks, JSX , props and state, Error boundary, ContextApi same as useContext but I use Redux, prop drilling i.e. passing props in Child Components, Pure components, Status codes, What if we update state directly?, use of render() in react [class based comp].

• ReactJS is a JavaScript library for building user interfaces, particularly for single-page applications where the content can be updated without requiring a full page reload. We use Components to build the apps. Components are the building blocks of React applications. They represent reusable and independent pieces of the user interface. Components can be either class-based or function-based.

• Virtual DOM: React keeps a lightweight representation of the real DOM in the memory, and that is known as the virtual DOM. When the state of an object changes, virtual DOM changes only that object in the real DOM, rather than updating all the objects.

• High performance: React updates only those components that have changed, rather than updating all the components at once. This results in much faster web applications.

• A higher-order component acts as a container for other components. This helps to keep components simple and enables re-usability. They are generally used when multiple components must use a common logic. This can be used with Error Boundary component by wrapping the App component in Error Boundary component.

• Parent and child components are uncontrolled & controlled comp respectively. i.e child comp is controlled by props being passed. When prop change, comp changes.

•  **React.StrictMode** - A React feature for identifying potential problems in your React application during development.

•  **useState** is a React Hook that allows functional components to manage state. It provides a way to declare state variables in function components, enabling them to maintain and update their state. What if we update state directly ?

• **useEffect** is used to perform side effects in functional components. It takes a function as its first argument, which will be executed after the component has rendered. The second argument is an array of dependencies, and the effect will re-run only if any of these dependencies change.

• **useRef** creates mutable reference. It doesn't cause re-render and gives reference to other obj/component. useRef with reference to Enter Key working in forms.

• **useContext** 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.

• **useReducer**(<reducer>, <initialState>)

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

• Synthetic events combine the response of different browser's native events into one API, ensuring that the events are consistent across different browsers.

• **React Router** is a library that enables navigation and routing in React applications. It allows you to create a single-page application with navigation between different views or components.

• **Code splitting** is a technique used in web development to improve the performance of a web application by breaking up large bundles of JavaScript code into smaller, more manageable pieces that can be loaded on demand. Instead of loading the entire JavaScript bundle when the application starts, only the necessary code for the current page or feature is loaded initially, and additional code is fetched as needed. Ex - Suspense and Lazy

• Drawbacks of using Reactjs.

**REDUX**

• Basics & core principles of redux - store, action and reducers. Redux is a state management library for JavaScript applications, commonly used with React. It helps manage the state of an application in a predictable way, making it easier to debug and understand. The Redux store is a single source of truth for the state of the entire application. Reducers are functions that specify how the application's state changes in response to actions sent to the store. The store holds the state and applies reducers to update the state.

• can we have multiple stores in redux ?

• Redux func - useSelector & useDispatch & connect

• Store: Holds the state of the application.

• Action: The source information for the store.

• Reducer: Specifies how the application's state changes in response to actions sent to the store.

• How is it beneficial to call API in redux than Components ?

**LifeCycleMethods for Class Based Components:**

• getInitialState(): This is executed before the creation of the component.

• componentDidMount(): Is executed when the component gets rendered and placed on the DOM.

• shouldComponentUpdate(): Is invoked when a component determines changes to the DOM and returns a “true” or “false” value based on certain conditions.

• componentDidUpdate(): Is invoked immediately after rendering takes place.

• componentWillUnmount(): Is invoked immediately before a component is destroyed and unmounted permanently.

• Mounting updating and unmounting using useEffect - Mounting just a function with empty dependency, update with dependency, Unmounting with return method in place of function without dependency.

**Request methods: What are the most common HTTP methods?**

• GET: Retrieves data on a server.

• POST: Sends new data to an API.

• PUT: Replaces an existing resource with an updated version.

• PATCH: Updates an existing resource, but does not require sending the entire body with the request.

• DELETE: Removes data from a database.

**Status codes** are standard codes returned by a server in response to a client's request made to a web server. These codes provide information about the status of the request and whether it was successful, encountered an error, or requires further action.

Here are some common HTTP status codes and their meanings:

- \*\*1xx Informational\*\*: These status codes indicate that the request has been received and the server is continuing the process.

- 100: Continue

- 101: Switching Protocols

- \*\*2xx Success\*\*: These status codes indicate that the request was received, understood, and processed successfully.

- 200: OK

- 201: Created

- 204: No Content

- \*\*3xx Redirection\*\*: These status codes indicate that further action needs to be taken to complete the request.

- 301: Moved Permanently

- 302: Found (Moved Temporarily)

- 304: Not Modified

- \*\*4xx Client Error\*\*: These status codes indicate that the request contains incorrect syntax or cannot be fulfilled.

- 400: Bad Request

- 401: Unauthorized

- 404: Not Found

- 403: Forbidden

- \*\*5xx Server Error\*\*: These status codes indicate that the server failed to fulfill a valid request due to an error on the server's side.

- 500: Internal Server Error

- 502: Bad Gateway

- 503: Service Unavailable

These are just a few examples, and there are many more status codes defined by the HTTP standard. Each status code provides specific information about the outcome of the request, allowing clients and servers to communicate effectively about the state of the request and any issues encountered.

Hypertext Transfer Protocol (HTTP) is a protocol that defines the rules for communication between a client and a server. Hypertext Transfer Protocol Secure (HTTPS) is a more secure version of HTTP

**NodeJS**

• Built in modules like fs, http, url, path, express etc

• Stream - Used to handle reading/writing data in chunks, making them efficient for processing large files or data.

• app.use & app.get

**MongoDb**

• Schema-less, key-value pair, objects, Document oriented (BSON), CRUD, Aggregation, Replica set & Sharding, Embedded & Reference Docx.

**NextJS**

• Provides additional features of server-side rendering SSR, static site generation SSG, built-in routing, and API handling, making it easier to build highly performant and SEO-friendly web applications.

• SSG - Next.js can generate static pages at build time, which can be served very quickly without server-side computation on each request.

• SSR is a powerful rendering technique where the server generates the HTML content of a web page and sends it to the browser, improving initial load time and SEO

• No need of Routing.