M.E.R.N

MongoDB

Beginner Interview Questions

What are the key features of MongoDB?

Answer:

- **Document-Oriented Storage:** Data is stored in flexible, JSON-like documents
- **Scalability:** Built to scale horizontally using sharding.
    - Sharding is a method for distributing data across multiple machines.
- **Indexing:** Supports various types of indexes to improve query performance.
    - **Single field indexes** These indexes collect and sort data from a single field in each document. They can be ascending or descending.
    - **Compound indexes** These indexes collect and sort data from multiple fields in each document. The order of the fields is important, and MongoDB recommends following the ESR rule: Equality, Search, Range.
    - **Multikey indexes** These indexes collect and sort data stored in arrays.
    - **Geospatial indexes** These indexes improve performance for queries on geospatial coordinate data.
    - **Text indexes** These indexes support text search queries on fields containing string content.
    - **TTL indexes** These indexes automatically remove documents from a collection after a certain amount of time.
- **Aggregation Framework:** Provides powerful ways to process and analyze data.
    - Rather than using **find()** you can build a "pipeline" of operations on your data to perform analytic operations
- **Replication:** Offers high availability through replica sets.
- Does MongoDB support foreign key constraints? No, MongoDB does not support foreign key constraints.
- **8. How do you perform a basic query in MongoDB?**
    - `db.collection.find({ name: "Alice" })`
- Aggregation
    - example, to calculate the average age of users in a collection:
        - `db.collection.aggregate([ { $group: { _id: null, averageAge: { $avg: "$age" } } } ])`

        - `This groups all documents and calculates the average of the age field!`

Advanced Questions

- **Data Aggregation**
    - Perform joins using the `$lookup` aggregation operation.
    - Sort and paginate data
    - Understand these methods
        - `$match`
        - `$count`
        - `$lookup`
        - `$unwind`
        - `$sort`
        - `$project`
        - `$limit`
        - `$merge`
        - `$facet`

Express.js

Beginner Questions

- **What is middleware in Express.js?**
    - **Answer:** Middleware in Express.js is a function that executes during the request-response cycle. It has access to the request and response objects and can modify them, end the request-response cycle, or call the next middleware function in the stack. Middleware functions can perform tasks such as logging, authentication, and request parsing

- **What is `req` and `res` in Express.js?**
    - `req` (request) represents the HTTP request and contains properties such as `req.body`, `req.params`, `req.query`, and `req.headers`.
    - `res` (response) is an object representing the HTTP response that will be sent to the client. It includes methods such as `res.send()`, `res.json()`, and `res.status()` for sending responses and setting HTTP status codes.
- What is the app.use used for?
    - the `app.use()` function is used to define middleware functions that are executed for every request to the application.
- Understand what npm do.

React JS

- **What is a component in React?**
    - a component is a JavaScript function or class that optionally accepts inputs (known as "props") and returns a React element that describes how a section of the UI should appear. Components can be functional or class-based. They are the building blocks of a React application.
- **What are props in React?**
    - Props (short for properties) are read-only attributes passed from a parent component to a child component. They are used to pass data and event handlers down the component tree. Props allow components to be dynamic and reusable.
- **React Hooks?**
    - **`useState()`:** Manages state in a functional component.
    - **`useEffect()`:** Handles side effects like data fetching or subscribing to external data sources.
    - **`useContext()`:** Accesses context values.
- What is a higher-order component (HOC) in React?
    - its a function that takes a component and returns a new component with additional props or functionality. HOCs are used for code reuse and to encapsulate logic that can be shared across multiple components.

- What is the virtual DOM in React?
    - The virtual DOM is a lightweight in-memory representation of the actual DOM. React maintains this virtual DOM to efficiently update the user interface

Advanced question

How can you optimize performance in a React application?

Several techniques can be used to optimize React application performance:

1. Memoization:
    - Use `React.memo` for functional components to prevent unnecessary re-renders.
    - Use `useMemo` and `useCallback` to memoize values and functions.
2. Code Splitting:
    - Implement code splitting using `React.lazy` and `Suspense` to load components only when needed.
3. Virtualization:
    - Use libraries like `react-window` or `react-virtualized` to render only visible items in large lists.
4. Avoid Inline Functions and Objects:
    - Inline functions and objects create new references on each render. Use `useCallback` and `useMemo` to avoid this.
5. Optimize Component Updates:
    - Use `shouldComponentUpdate` or `PureComponent` in class components, and `React.memo` for functional components to optimize rendering.
6. Efficient Reconciliation:
    - Use proper keys in lists to help React identify which items have changed.
7. Avoid Expensive Computations:
    - Perform heavy computations outside of render or use web workers if necessary.
8. Server-Side Rendering (SSR):
    - Consider using SSR to send pre-rendered HTML to the client, improving load times and SEO.

Node js

What do you mean by Asynchronous API?

All APIs of Node.js library are aynchronous that is non-blocking. It essentially means a Node.js based server never waits for a API to return data. Server moves to next API after calling it and a notification mechanism of Events of Node.js helps server to get response from the previous API call.

What's the difference between operational and programmer errors?

Operation errors are not bugs, but problems with the system, like request timeout or hardware failure. On the other hand programmer errors are actual bugs.

What is control flow function?

It is a generic piece of code which runs in between several asynchronous function calls is known as control flow function.

What are Event Listeners?

Event Listeners are similar to call back functions but are associated with some event.

Node.js has built in event's and built in event listeners. Node.js also provides functionality to create Custom events and Custom Event listeners.

How you can monitor a file for modifications in Node.js ?

We can take advantage of File System watch() function which watches the changes of the file.

Advanced Questions

What are the global objects of Node.js?

- process - The process object is a global that provides information about, and control over, the current Node.js process.
- console - Used to print to stdout and stderr.
- buffer - Used to handle binary data.

When to not use Node.js?

We can use Node.js for a variety of applications. But it is a single threaded framework, so we should not use it for cases where the application requires long processing time. If the server is doing some calculation, it won't be able to process any other requests. Hence, Node.js is best when processing needs less dedicated CPU time.

How do you debug Node.js applications?

Node has its own built in GUI debugger as of version 6.3 (using Chrome's DevTools).

```
node --inspect server.js
```

ASP.NET

Describe the ASP.NET page life cycle.

Stages of the life cycle is from initialization to loading and rendering, and finally disposal.

Before learning anything you must understand

- Front end
    - What is local storage
    - Cookies
    - Query strings

- Backend
    - Http Context: The HttpContext class offers an intuitive way to access request and session state. Further, it inherits from HttpContextBase, which is testable in isolation, aiding in unit testing.

Key Events in the Page Lifecycle Page

- Request: This sets in motion the entire page
- lifecycle. Start: The page's structure is established.
- Initialization: Components, such as master pages, are initialized.
- Load: Data binding and loading occur.
- Postback: Handles form submissions and validations.
- Rendering: The page is transformed to HTML.
- Unload: The page is disassociated from the server.

Understanding the Stages

- PreInit: During this step, the page initializes components like master pages and themes.
- Init: The page sets properties that might be modified or reset later.
- InitComplete: Any post-init tasks are completed.
- PreLoad: Actions that need to be completed before the page and its controls are loaded can be executed during this step.
- LoadComplete: After this stage, all page life cycle events might have completed.
- PreRender: Actions before the page or any of its controls are rendered are performed here.
- SaveStateComplete: ViewState, form data, and other data relevant to the page are saved. After this stage, if a postback is performed, any data from the current page instance is lost, and the page reverts to the version that was saved during this lifecycle stage.

What are WebForms in ASP.NET?

WebForms, a core feature of ASP.NET, enable rapid web application development through a variety of visual tools. These tools offer an intuitive method to build rich, interactive web applications while handling low-level plumbing tasks automatically. WebForms fundamentally abstract the stateless nature of the web.

They encapsulate web pages as stateful entities, mirroring desktop application behavior.

Understand Central Componenets

- Pages
- User Controls
- Master Pages
- State Management
- Event Cycle

- Visual studio integrations

Describe the role of a master page in ASP.NET WebForms.

Key Feature

- Consistent Layout: Master Pages ensure uniform design across web pages.
- Separation of Concerns: Division between Master Pages and content pages allows independent editing or updating.
- Selective Inheritance: Different content pages can leverage distinct Master Pages. UI
- Core Elements: Master Pages provide essential elements such as headers, footers, navigation, and placeholders.

When to Use Master Pages

- Corporate Branding: Employ a consistent design theme reflecting the corporate identity.
- Large Applications: For simplified maintenance and updates in multi-page apps.
- Centralize Management: Where core design features or controls need to be managed from a single point.
- Uniformity: To ensure visual consistency throughout the application

Custom control in ASP.NET

- **Control Design**: Create the visual representation of your control using the designer or directly building the HTML in your .aspx file.
- **Code-Behind Logic**: Use the .ascx.cs file to implement the control's functionalities, such as event handling or complex UI manipulations.
- **Control Packaging**: The .ascx file and its code-behind form the control, which you can distribute as a reusable unit.

Code-Behind and Control Reference Approach

- Control Design: Call Controls.Add() method in the .ascx.cs file to create a control and add it to the control's hierarchy. This is done in lieu of a visual design in the .ascx file itself.
- Control Packaging: Just like the Visual and Code-Behind method, this one sees distribution in a packaged format.

Best Practices for Building Custom Controls

- Separation of Concerns: For better maintainability, keep your control's presentation logic in .ascx, and the business and event handling logic in the .ascx.cs.
- Custom Control vs. User Control: Both custom controls and user controls can provide modularity. User controls are easier to build, while custom controls offer more fine-tuned control. The choice depends on your project's requirements and your development team's expertise. User controls are easier for separating concerns.
- Documentation: Whether you opt for a code-behind approach or the control reference method, your control must be well-documented for reusability.