1. **What are middleware functions in Express.js, and how do they work?**

Middleware functions in Express.js are functions that have access to the request object “req”, the response object “res”, and the next middleware function in the application’s request-response cycle. These functions can make changes to the request and response objects, call the next middleware function, or end the request-response cycle. The primary roles of middleware include

- 1. **Request Logging**: Log details of the request.
- 2. **Body Parsing**: Parse incoming request bodies.
- 3. **Authentication**: Determine if the user is logged in.
- 4. **Authorization**: Check if the logged-in user has permission to access a specific resource.

Middleware functions are invoked sequentially, and each function can stop the chain by not calling next() or can pass control to the next middleware function by calling next().

2. **What is jwt, and how does it work?**

jwt is like a secure pass that allows client and server to exchange information safely over the internet. jwt has three parts: 1. Header (tells type of jwt and type of secure method like RSA, SHA256, etc), payload (actual data being exchanged), and signature (verifies token comes from a trusted source and no one altered it in exchange process, created by combining secret key with header and payload in specific algorithms).

3. **How do you securely store jwt on the client-side?**

First, ensure all communications between the client and server are encrypted with Https to prevent interception by attackers. Second, store jwt in HttpOnly Cookie to prevent cookies from being accessed via client-side scripts to against cross-site scripting (XSS) attacks. Third, use the Secure attribute on cookies to ensure they are only transmitted over secure (HTTPS) connection. Fourth, implement expiration times for tokens to manage token renewal to reduce impact of potential token theft.

4. **How does token expiration work in JWT?**

jwt often include an expiration time (exp) in their payload, which tells how long the token is valid. When jwt is used in an authentication request, the server checks this ‘exp’ to decide if the current time has passed the specified expiration time. If passed, the token is considered expired and the server rejects the authentication request, often prompting the user to log in again or refresh the token. This helps to limit the duration for which a stolen or leaked token can be used.