Answers.

1.What is client-side and server-side in web development, and what is the main difference between the two?

Answer: The main difference is that client-side code is responsible for the presentation and interaction with the user interface, while server-side code handles data processing, business logic, and interacts with databases.

2.What is an HTTP request and what are the different types of HTTP requests?

Answer: An HTTP request is a message sent by a client (such as a web browser) to a web server, specifying an action to be performed. It consists of a request method, URL, headers, and an optional request body. The server processes the request and returns an HTTP response with the requested data or performs the requested action.

The different types of HTTP requests include:

GET, POST, PUT, DELETE, PATCH, HEAD, OPTIONS

3.What is JSON and what is it commonly used for in web development?

Answer: JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is commonly used for storing and exchanging structured data between a client and a server in web development. JSON is easily readable by humans and can represent data in the form of key-value pairs or arrays.

4.What is a middleware in web development, and give an example of how it can be used.

Answer: In web development, middleware refers to functions or components that sit between the web server and the application's request/response handling logic. It can perform various tasks such as request preprocessing, authentication, logging, and more. For example, a middleware for authentication can intercept incoming requests, check if the user is authenticated, and either allow or deny access to protected resources based on the authentication status.

5.What is a controller in web development, and what is its role in the MVC architecture?

Answer: In the Model-View-Controller (MVC) architectural pattern, the controller is responsible for handling user input, processing requests, and generating appropriate responses. It acts as an intermediary between the model (which represents data and business logic) and the view (which handles the presentation layer). The controller receives input from the user, interacts with the model to fetch or update data, and then determines the appropriate response, often by invoking the appropriate view for rendering. It plays a crucial role in separating concerns and maintaining the flow of data and actions within an MVC application.