Part-1

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

Ans:- The main difference between client-side and server-side is the location where the code is executed. Client-side code runs on the user's device, while server-side code runs on the web server. Client-side code is responsible for the user interface and user interaction, while server-side code handles data processing, storage, and business logic. The client-side code is easily accessible and modifiable by users, while server-side code remains hidden and secure on the server.

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

Ans: An HTTP request is a message sent by a client (typically a web browser) to a web server to initiate a specific action or retrieve information. It follows the rules and protocols of the Hypertext Transfer Protocol (HTTP), which is the foundation of data communication on the World Wide Web.

The different types of HTTP requests are:

1. **GET**: Retrieves a resource or data from the server. The requested data is included in the URL query parameters.
2. **POST**: Submits data to be processed by the server. The data is sent in the body of the request and is often used for form submissions, file uploads, or creating new resources.
3. **PUT**: Updates an existing resource with new data. The entire resource is usually sent in the body of the request.
4. **DELETE**: Requests the server to delete a resource specified in the URL.
5. **PATCH**: Partially updates an existing resource with new data. Only the specified changes are sent in the body of the request.
6. **HEAD**: Similar to a GET request, but it retrieves only the HTTP headers of a resource, without the actual content. It is often used to check the status or metadata of a resource.
7. **OPTIONS**: Retrieves the HTTP methods supported by a server for a specific resource. It helps determine which methods can be used for subsequent requests.
8. **TRACE**: Echoes back the received request to the client, allowing it to see what the intermediate servers modified in the request. It is primarily used for debugging and diagnostics.

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

Ans: JSON (JavaScript Object Notation) is a lightweight data interchange format that is easy for humans to read and write and easy for machines to parse and generate. It is based on a subset of

the JavaScript programming language, but it is language-independent and widely used in web development.

JSON is commonly used in web development for various purposes:

1. **Data interchange**
2. **API communication**
3. **Configuration files**
4. **Storage of structured data**
5. **Client-side data manipulation**
6. **AJAX requests**

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

Ans: Middleware is a (loosely defined) term for any software or service that enables the parts of a system to communicate and manage data. It is the software that handles communication between components and input/output, so developers can focus on the specific purpose of their application.

Let's make an example.

You have a request coming to your server and you're about to process it and you need to go through a few steps, for instance, add the Set-Cookie header, make some logs and apply a template to get the HTML to respond with.

Instead of doing all those things inside a single huge handler function just split them into different functions like this:

```
1. // the request starts here, ctx is a special object which has all the
   initial data
2.
3. setCookieHeader(ctx);
4. writeLogs(ctx);
5. applyTemplate(ctx);
6.
7. // ctx has all the data changed properly, it's time to respond back to
   the client
```

These separate functions are **middlewares**.

So, in general, it's a term for any software or service that enables the parts of a system to communicate and manage data.

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

Ans: In web development, a controller is a component or module that handles the user's requests and manages the flow of data between the model and the view in the MVC (Model-View-Controller) architecture. The primary role of a controller is to act as an intermediary between the user interface (view) and the data (model).

The specific role of a controller can vary depending on the framework or technology being used, but generally, it performs the following tasks:

1. Receives user input
2. Processes user
3. Interacts with the model
4. Updates the view
5. Manages the flow