1. Difference between inline, block, and inline-block

- **Inline**: Elements do not start on a new line and only take up as much width as necessary. Examples: <span>, <a>. We can't set width or height on these directly.
- **Block**: Elements start on a new line and take up the full width of their container by default. Examples: <div>, <p>. We can style their width, height, margin, etc.
- **Inline-block**: It behaves like inline in that it doesn't start on a new line, but you can set width, height, and box model properties like padding and margin. It's like a mix of the two.

2. What is the CSS Box Model? Explain its parts.

The CSS Box Model describes how every element on a webpage is structured. It's made up of the following parts:

1. **Content**: The actual content of the box, like text or an image.
2. **Padding**: Space between the content and the border. It adds spacing *inside* the element.
3. **Border**: A line surrounding the padding (if present). You can style it (e.g., thickness, color).
4. **Margin**: The space between the element's border and other elements on the page.

3. Difference between GET and POST requests

- **GET**:
  - Used to *retrieve* data from the server.
  - Parameters are appended to the URL, making it less secure .
  - Cacheable, lightweight, and typically used for requests that don't modify server data.
- **POST**:
  - Used to *send* data to the server to create/update resources.
  - Parameters are included in the request body, making it more secure for sensitive data.
  - Not cached, and used when actions affect server state (like submitting forms).

4. How to handle CORS errors in API calls?

CORS (Cross-Origin Resource Sharing) errors occur when your frontend and backend are on different origins, and the server doesn't explicitly allow the browser to make the request. Here's how to handle it:

1. **Modify the Server**:

   Configure the backend to include the Access-Control-Allow-Origin header with the allowed origins or use ' * ' to allow all origins. Example for Express.js:

```
app.use((req, res, next) => {
  res.header("Access-Control-Allow-Origin", "*");
  res.header("Access-Control-Allow-Methods", "GET, POST, PUT, DELETE");
  res.header("Access-Control-Allow-Headers", "Content-Type");
  next();
});
```

2. **Use a Proxy**:

   Set up a proxy in your frontend development environment (e.g., in React, use the proxy field in package.json).

3. **Enable CORS on the API Gateway**:

   If using cloud services (e.g., AWS, Azure), configure CORS in the API gateway settings.

4. **Avoid CORS by Same-Origin**:

   Host the frontend and backend on the same domain if possible.