CORS (Cross-Origin Resource Sharing) - Full Lesson

1. **Definition:** CORS is a security feature implemented by web browsers that restricts web pages from making requests to a different domain than the one that served the web page. This is part of the browser's **same-origin policy**.

2. **Why CORS Exists:** Browsers prevent malicious websites from reading sensitive data from another site without permission. Without CORS, a site could fetch data from another site using your credentials without you knowing.

3. **Same-Origin Policy:** Two URLs have the same origin if they have the **same protocol, host, and port**. Example:

4. Allowed: `http://example.com/page1` and `http://example.com/page2`

5. Blocked: `http://example.com` and `http://api.example.com` (different host)

6. **How CORS Works:** When a browser requests a resource from another origin, it may send a **preflight request** (OPTIONS request) to check if the server allows the request.

The server responds with headers like: - `Access-Control-Allow-Origin`: Which origin is allowed. - `Access-Control-Allow-Methods`: Which HTTP methods are allowed (GET, POST, etc). - `Access-Control-Allow-Headers`: Which headers the request can include. - `Access-Control-Allow-Credentials`: Whether cookies/auth headers are allowed.

Example headers:

```
Access-Control-Allow-Origin: https://example.com
Access-Control-Allow-Methods: GET, POST, PUT
Access-Control-Allow-Headers: Content-Type, Authorization
Access-Control-Allow-Credentials: true
```

1. **Backend Implementation:**

2. **Node.js (Express):**

```
const express = require('express');
const cors = require('cors');
const app = express();

app.use(cors({ origin: 'https://example.com', credentials: true }));
```

3. **C#/.NET:**

```
services.AddCors(options => {
    options.AddPolicy("AllowExample", builder => {
        builder.WithOrigins("https://example.com")
               .AllowAnyMethod()
               .AllowAnyHeader()
               .AllowCredentials();
    });
});

app.UseCors("AllowExample");
```

4. **Common Issues:**

5. Using `*` with credentials is **not allowed**.
6. Preflight OPTIONS requests must be handled correctly.

7. Only browsers enforce CORS; server-to-server requests are not restricted.

8. **Summary:** CORS is about **controlled cross-origin access**. As a backend dev, your role is to **set the right headers** so browsers allow legitimate requests while maintaining security.