# Calling External API

This is a simple Node.js + Express API that acts as a proxy to generate QR codes using the goqr.me external API. It demonstrates how to integrate external APIs into your own server using only native Node.js modules.

## What you'll get to know

- How to run a Node.js/Express server
- How to proxy requests to external APIs
- Working with binary data (images)
- Using native Node.js `https` module for HTTP requests
- Setting response headers for images
- Input validation and error handling
- Request logging middleware

## Prerequisites

- Node.js (recommend v20+)
- npm (comes with Node)

## Quick start

1. Install dependencies

```
npm install
```

2. Start the server

```
npm start
# Or you can start directly using node:
node server.js
```

3. Open the API in your browser: `http://localhost:3000`

## Endpoints

### Health Check

- **GET /health** — simple health check, returns server status and uptime

### QR Code Generation (External API Proxy)

- **GET /qr** — generate a QR code image
    - **Query Parameters:**
        - `data` (required): The text/URL to encode in the QR code
        - `size` (optional): Size in pixels (50-1000, default: 200)
    - **Returns:** PNG image that can be displayed in browsers or saved

### Example Responses

**GET /health**

```
{
  "status": "ok",
  "uptime": 123.456
}
```

**GET /qr?data=Hello%20World&size=300**

Response: PNG image (binary data)

You can open this directly in a browser to see the QR code image!

**Error Response (missing data parameter)**

```
{
  "error": "data parameter is required",
  "usage": "GET /qr?data=YourTextHere&size=200"
}
```

## Features

- **Proxy Pattern**: Server acts as a middleman between client and external goqr.me API
- **Native Node.js**: Uses built-in `https` module (no axios or other HTTP libraries needed)
- **Input Validation**: Validates data and size parameters
- **Request Logging**: All requests are logged to the console with timestamps
- **Error Handling**: Proper HTTP status codes and error messages
- **CORS Support**: Allows cross-origin requests for frontend integration
- **Image Streaming**: Pipes image data directly from external API to client
- **Caching**: QR codes are cached in the browser for 1 day

## Try it with curl (Windows PowerShell examples)

```
# Check server health
curl http://localhost:3000/health | ConvertFrom-Json

# Generate a QR code and save it
curl http://localhost:3000/qr?data=HelloWorld&size=300 -OutFile qrcode.png

# Generate QR code with URL
curl http://localhost:3000/qr?data=https://github.com&size=400 -OutFile github-qr.png

# Or just open in browser to see it
Start-Process "http://localhost:3000/qr?data=https://github.com&size=400"
```

## Try it in the browser

Simply paste these URLs in your browser:

- `http://localhost:3000/health` — Check server status
- `http://localhost:3000/qr?data=Hello%20World` — Generate QR code with text
- `http://localhost:3000/qr?data=https://github.com&size=500` — Generate large QR code with URL

## What you can try next

1. Add rate limiting middleware to prevent abuse
2. Implement caching on the server side to reduce external API calls
3. Add support for different QR code formats (SVG, PDF)
4. Create an HTML frontend with a form to generate QR codes
5. Add more external API integrations (weather, jokes, etc.)
6. Implement analytics to track popular QR code requests
7. Add authentication to protect the endpoint

## Technical Notes

- Uses native Node.js `https` module instead of external HTTP libraries
- The server doesn't store QR codes, it streams them directly from the external API
- Response headers include `Cache-Control` to cache images in the browser for 24 hours
- The external API used is: `api.qrserver.com`
- Images are piped directly from the external API to the client for efficiency