SQL in our App

Module 5 Week 11

Notes Repo: https://github.com/C-Shi/lhl-flex-lecture

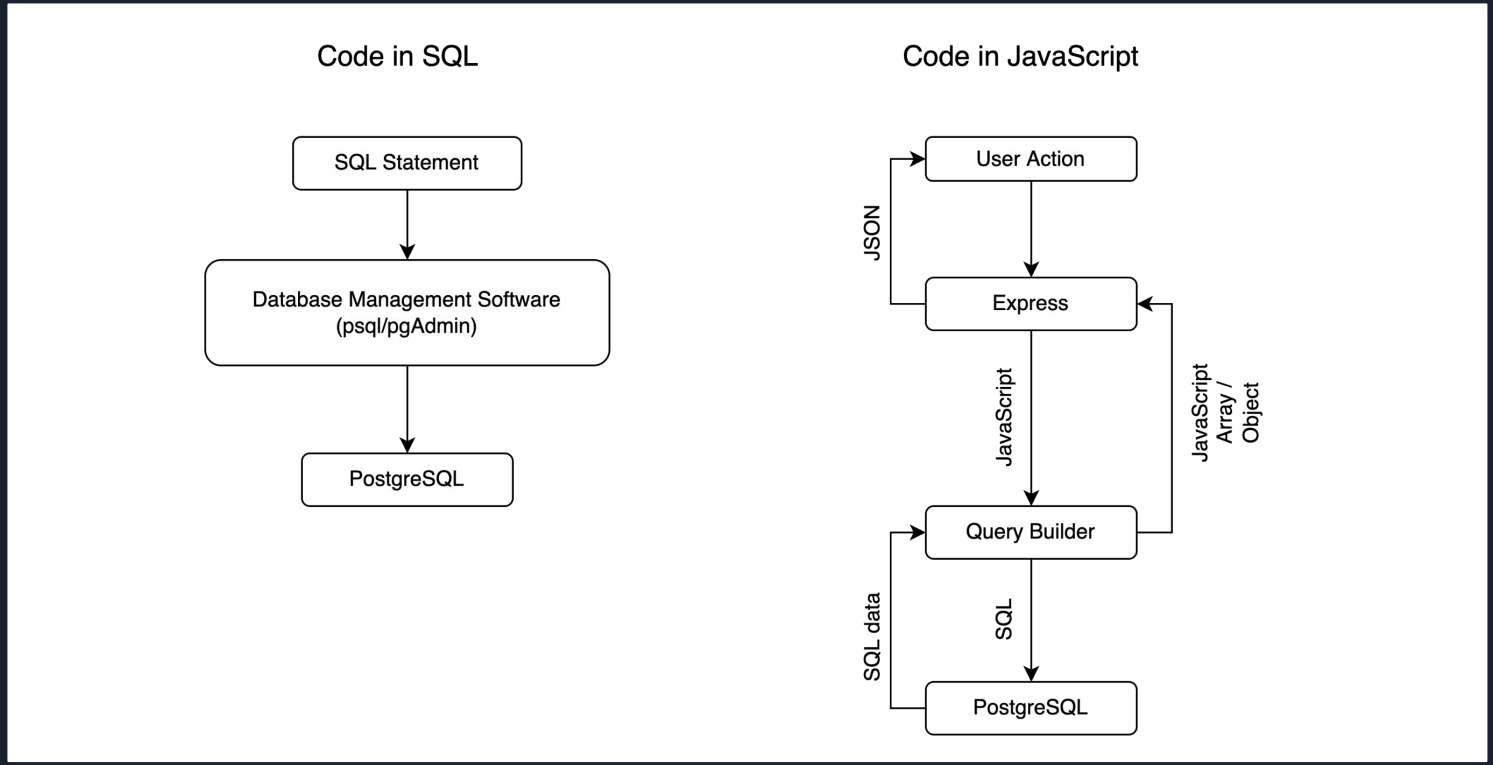
Learning Objectives

Understand how application interact with database

Using node-postgres to build a todo List

SQL and Web Security

Interacting with SQL Database



node-postgres

An npm package to interfacing with PostgreSQL database

Support callback and promise, we will use promise

It allows us to communicate with PostgreSQL in JavaScript

For example:

```
pool.query('SELECT * FROM students WHERE id = $1', [id])
  .then(result => {
    res.render('students', result.rows)
  })
```

SQL review

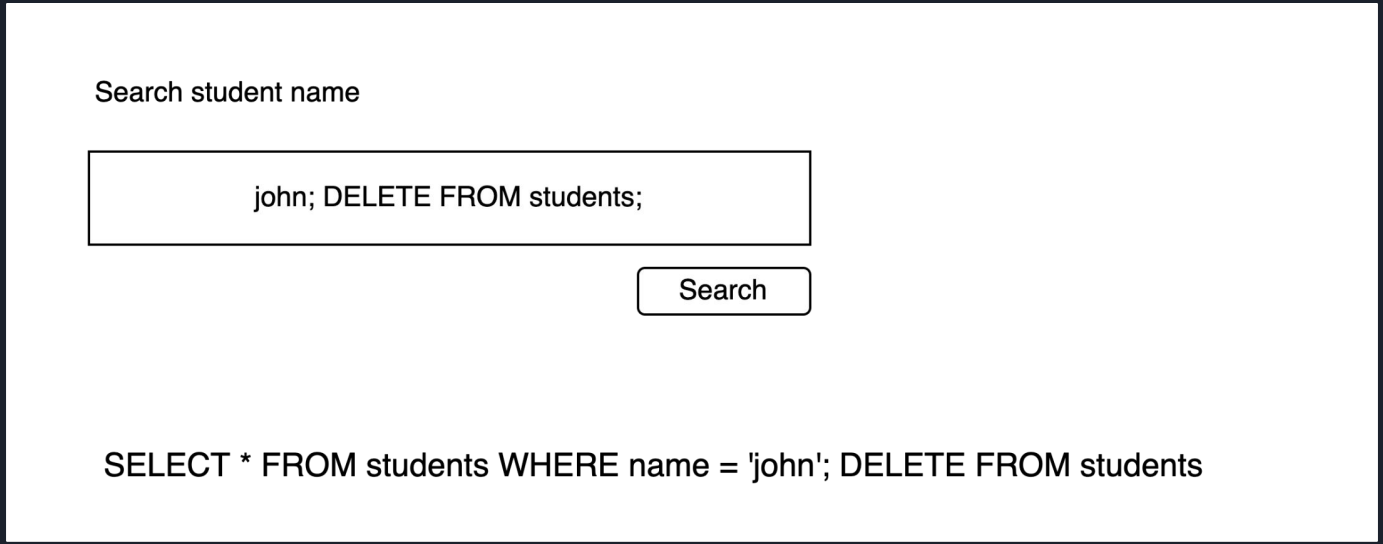
```sql
-- List all students
SELECT * FROM students;

-- Get student with id 1
SELECT * FROM students WHERE id = 1;

-- Create a student profile
INSERT INTO students (name, email, year, gpa) VALUES ('Jason', 'jason@gmail.com',
1, 3.5);

-- Update student 1's gpa to 3.0
UPDATE students SET gpa = 3.0 WHERE id = 1;

-- DELETE student who id is 10
DELETE FROM students WHERE id = 10;
```

Web Security - SQL Injection

A web security vulnerability that allows attackers to interfere with the queries

Directly inject user input into database as trusted SQL

Solution is to run query sanitation, use prepared statement



Web Security - Credentials

Credentials like database connection password SHOULD NOT be publically available

It can also changed based on environment. Eg: Dev, Production

Use environmental variable and ignore the file in GIT

Node has a global `process` object, which contains `env`

Use a package called dotenv

Summary

We code in JavaScript, but use **pg** translate our code into SQL

User input need to be sanitized. We use prepared statement

Credentials should not be tracked by GIT and can be stored as environment variables