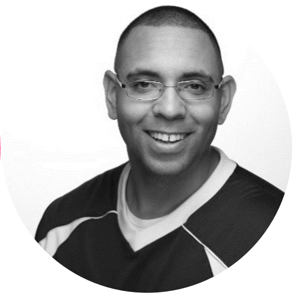
Coding Your Apps Functionality in JavaScript



Steve Buchanan
DevOps Architect

@buchatech | www.buchatech.com

Overview

Connecting our Back End JavaScript REST API to a Database

Utilizing Event Handlers In our JavaScript Web App

Building JavaScript Functions for CRUD Operations & Connecting our Front End to the REST API

Connecting our Back End JavaScript REST API to a Database

Database Hosting Options

render

Azure
MySQL
PostgresSQL

AWS
MySQL
PostgresSQL

GCP
MySQL
PostgresSQL

PostgresSQL
https://render.com/docs/databases

MySQL
https://render.com/docs/deploy-mysql

```sql
// Code for creating the MySQL Database
CREATE DATABASE jokesdb;

// Selecting the Database
USE jokesdb;

// Creating a table in your Database
CREATE TABLE jokes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  question VARCHAR(255),
  answer VARCHAR(255),
  rating INT
);
```

Build MySQL Database

Use SQL query language to create the database and the table

```sql
INSERT INTO jokes (question, answer, rating)
VALUES
  ("Why couldn't the bicycle stand up by itself?", "Because it was two-tired.", 0),
  ("I'm reading a book on anti-gravity.", "It's impossible to put down.", 0),
  ("Did you hear about the guy who invented Lifesavers?", "He made a mint.", 0),
  ("I used to be a baker, but I couldn't make enough dough.", "", 0),
  ("What do you call a pile of cats?", "A meowntain.", 0),
  ("What do you get when you cross a snowman and a vampire?", "Frostbite.", 0);
```

Build MySQL Database

Insert Jokes into the table to get started

Includes the joke question, answer, and rating of 0

The id will automatically be generated

Connect our REST API to our MySQL Database

dbinfo.js

```
// Telling JavaScript to require mySQL in the script
const mysql = require('mysql');

// The connection info for the database
const db = mysql.createConnection({
  host: 'mysqldb1.mysql.database.azure.com',
  user: ‘mydbadmin',
  password: ‘#1000joke$',
  database: 'jokesdb'
});

// Function for connecting to the database with error handling if
connection fails
db.connect((error) => {
  if (error) {
    console.error('Database connection error:', error);
  } else {
    console.log('Connected to the database');
  }
});

// Making the database connection available for use in other
JavaScript files
module.exports = db;
```

REST API Code

apiserver.js

```
const express = require('express');
const db = require('./config/dbinfo');
const router = express.Router();

// Create a joke
router.post('/jokes', (req, res) => {
  let joke = req.body;
  let sql = 'INSERT INTO jokes (question, answer, rating) VALUES (?, ?, ?)';
  let values = [joke.question, joke.answer, joke.rating];
  db.query(sql, values, (err, result) => {
    if (err) throw err;
    res.send('Joke has been added...');
  });
});

// Update a joke
router.put('/jokes/:id', (req, res) => {
  let joke = req.body;
  let sql = `UPDATE jokes SET question='${joke.question}', answer='${joke.answer}',
rating=${joke.rating} WHERE id=${req.params.id}`;
  db.query(sql, (err, result) => {
    if (err) throw err;
    res.send('Joke updated...');
  });
});

// Delete a joke
router.delete('/jokes/:id', (req, res) => {
  let sql = `DELETE FROM jokes WHERE id=${req.params.id}`;
  db.query(sql, (err, result) => {
    if (err) throw err;
    res.send('Joke deleted...');
  });
});

module.exports = router;
```

Utilizing Event Handlers In our JavaScript Web App

Event Handlers In JavaScript

Event Handling in JavaScript lets you handle user actions on the page, like mouse moves and key presses

Event handlers are JavaScript code that invoke code when an action happens on an HTML element or a DOM element

Event handlers can invoke direct JavaScript code or a function

A single HTML tag can have multiple event handlers & will invoke specific code depending on the event or action that happens on the HTML/DOM element

Event Handler for Retrieving a Random Joke

index.html

```html
<body onload="getRandomJoke()"> <!-- Call the function on page load -->
  <div class="jumbotron text-center">
    <h1 class="display-4">Dad Jokes</h1>
    <p class="lead">Get ready to laugh!</p>
    <hr class="my-4">
    <h3 id="joke-question"></h3>
    <p id="joke-answer"></p>
    <button class="btn btn-primary" onclick="getRandomJoke()">Get Another
Joke</button>
    <a href="/features/addjoke/addjoke.html" class="btn btn-success">Add a Joke</a>
    <a href="/features/jokelist/jokelist.html" class="btn btn-info">List all
Jokes</a>
  </div>
```

```html
  <script>
  // Function to get a random joke
  function getRandomJoke() {
      fetch('/japi/randomjokes')
        .then(response => response.json())
        .then(data => {
          const jokeQuestion = document.getElementById('joke-question');
          const jokeAnswer = document.getElementById('joke-answer');
          jokeQuestion.textContent = data[0].question;
          jokeAnswer.textContent = data[0].answer;
        })
        .catch(error => console.log(error));
    }
  </script>
```

Building JavaScript Functions for CRUD Operations & Connecting our Front End to the REST API

Connecting to the REST API

index.js

```
// Add middleware / dependencies
const express = require('express');
const db = require('./config/dbinfo');
const app = express();
const port = 80;
const jokeRouter = require('./apiserver');
app.use(express.json());
app.use(express.static(__dirname));

// Use API routes
app.use('/japi', jokeRouter);

// Entry point for the web app
app.get('/', (req, res) => {
    res.sendFile(__dirname + '/views/index.html');
  });

// 404 route
app.use((req, res) => {
  res.status(404).send(`
    <html>
      <head>
        <title>Page Not Found</title>
      </head>
      <body>
        <h1>The page you are trying to access does not exist.</h1>
        <p>Enjoy some jokes by going to the:</p>
        <p>Dad Jokes <a href="/">HOME</a>.</p>
      </body>
    </html>
  `);
});

// Start the Server
app.listen(port, () => {
    console.log('Server started on port 80');
  });
```

Create a Joke

```javascript
// Function to add a new joke
async function addJoke(question, answer, rating) {
  try {
    const response = await fetch('/japi/jokes', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({ question, answer, rating }),
    });

    if (response.ok) {
      alert('Joke added successfully!');
    } else {
      alert('Failed to add joke.');
    }
  } catch (error) {
    console.log(error);
  }
}
```

Read (Display) Single Joke

```
// A single joke
router.get('/japi/jokes/:id', (req, res) => {
  let sql = `SELECT * FROM jokes WHERE id = ${req.params.id}`;
  db.query(sql, (err, result) => {
    if (err) throw err;
    res.send(result);
  });
});
```

Read (Display) All Jokes

```
// Function to display all jokes
function displayJokes() {
    fetch('/japi/jokes')
      .then(response => response.json())
      .then(data => {
        const jokeList = document.getElementById('joke-list');
        jokeList.innerHTML = '';

        data.forEach(joke => {
          const listItem = document.createElement('li');
          listItem.className = 'list-group-item';
          listItem.innerHTML = `
            <strong>Question:</strong> ${joke.question}<br>
            <strong>Answer:</strong> ${joke.answer}<br>
            <strong>Rating:</strong> ${joke.rating}<br>
            <button class="btn btn-sm btn-primary" onclick="editJoke(${joke.id})">Edit</button>
            <button class="btn btn-sm btn-danger" onclick="deleteJoke(${joke.id})">Delete</button>
          `;
          jokeList.appendChild(listItem);
        });
      })
      .catch(error => console.log(error));
```

Edit a Joke

```
// Function to edit a joke
  function editJoke(id) {
    const newQuestion = prompt('Enter the new question:');
    const newAnswer = prompt('Enter the new answer:');
    const newRating = prompt('Enter the new rating:');

    const joke = {
      question: newQuestion,
      answer: newAnswer,
      rating: newRating
    };

    fetch(`/japi/jokes/${id}`, {
      method: 'PUT',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify(joke)
    })
      .then(response => response.text())
      .then(data => {
        alert(data);
        displayJokes();
      })
      .catch(error => console.log(error));
  }
```

Delete a Joke

```javascript
// Function to delete a joke
function deleteJoke(id) {
  if (confirm('Are you sure you want to delete this joke?')) {
    fetch(`/japi/jokes/${id}`, {
      method: 'DELETE'
    })
      .then(response => response.text())
      .then(data => {
        alert(data);
        displayJokes();
      })
      .catch(error => console.log(error));
  }
}
```

Demo

Demo: Running the Code

Summary

In this module we covered:

- ❑ Learned about Database hosting options
- ❑ Reviewed the code for creating the database, tables, and initial data
- ❑ Connected our REST API to our MySQL Database
- ❑ Connected our Frontend to our REST API
- ❑ Explored utilizing Event Handlers In our JavaScript
- ❑ Updated our HTML code and built JavaScript for CRUD operations

Why this is important:?

Most web apps have data and therefore will need a database to store that data. Its important to know how to work with databases & be able to connect to them from JavaScript code.

Also, with most web apps they will be split into multiple services via REST API's its important to know how to work with REST APIs with JavaScript based web apps.

Once you have a database and a REST API in place you need to have forms and functions for input that take in the data, work with it and read/write to the database.