LU10c - Node.js - Under Construction

Learning Objectives

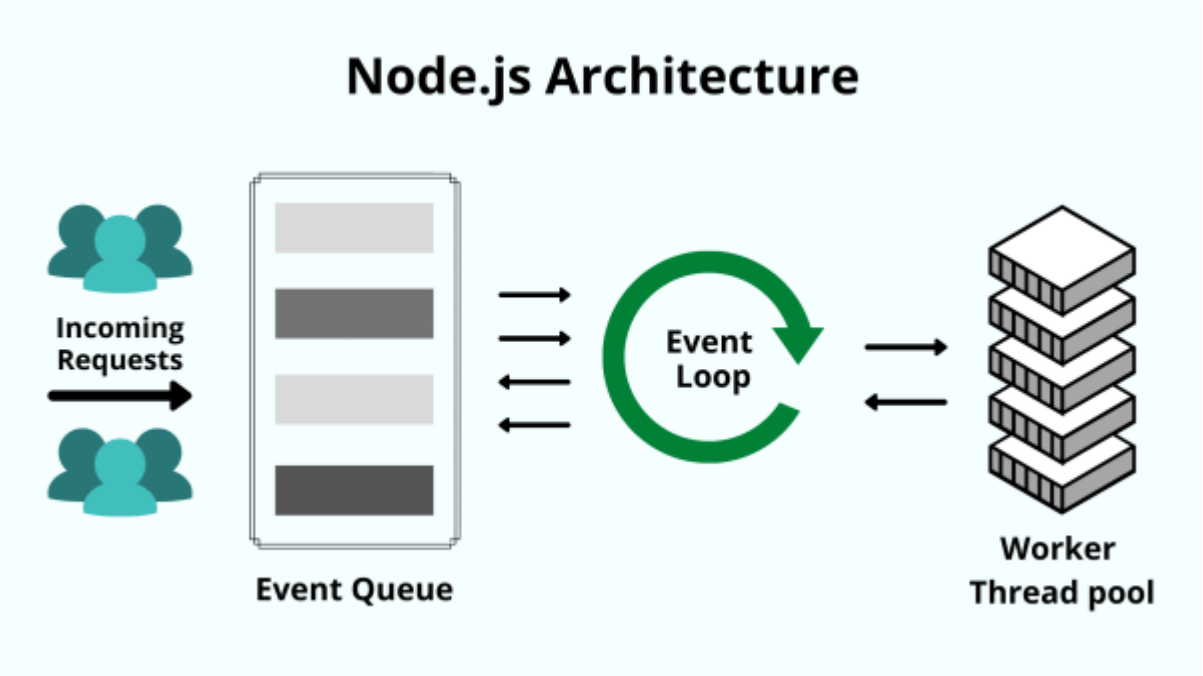
1. What a Node.js is
2. Which benefits a node server provides us
3. programm a node server

What is Node.js?

A Node.js server is a backend system built using Node.js, a runtime environment that allows you to write server-side applications in JavaScript. This server can handle incoming requests, process data, communicate with databases, and return responses to the client (such as a web browser or mobile app). Here’s a breakdown of how a Node server works and why it’s popular:

1. **JavaScript on the Server-Side:** Traditionally, JavaScript was only used in browsers for frontend development. Node.js allows JavaScript to be run on the server side, meaning that developers can use one language across the full stack (both front and back end).
2. **Asynchronous, Non-Blocking:** Node.js uses an event-driven, non-blocking I/O model, which is well-suited for handling many concurrent requests. Instead of waiting for one operation to finish before starting another (as in synchronous processing), Node’s asynchronous model can handle multiple tasks at once, making it faster and more efficient, especially for real-time applications.
3. **Single-Threaded with Event Loop:** Unlike traditional multi-threaded servers, a Node server is single-threaded but uses an event loop to manage tasks. This architecture allows it to efficiently manage and prioritize incoming tasks without creating multiple threads, which reduces overhead and increases performance.
4. **Wide Usage in Real-Time Applications:** Node servers are commonly used in applications that require real-time data, like chat applications, online gaming, collaborative tools, and streaming services. Its efficient handling of multiple, simultaneous connections makes it ideal for these scenarios.
5. **Ecosystem and Libraries:** Node.js has a large ecosystem of modules available through npm (Node Package Manager). Popular frameworks, such as Express.js, provide structured tools and libraries for creating robust, scalable APIs and applications.

In short, a Node.js server provides a fast, efficient way to handle web requests, especially for applications that require real-time interactions and high scalability.



Our first node server

```
/*
**********************************************************************************
***************
* Author: V. Demir, 30.10.2024
*
**********************************************************************************
***************
* Description:
* Express-Server, that takes a request from the browser, fetches the result
from the
* database, and provides the browser with the requested data.
*
**********************************************************************************
***************
* Hints:
* Before programming the server the following list of JS-Modules
* must be imported to our IDE. Execute the commands below in the terminal!
* npm install node
* npm init -y
* npm install mysql
* npm install body-parser
* npm install express
**
**********************************************************************************
************* */
// Reference: www.npmjs.com/package/mysql
```

```
const mysql = require("mysql"); // import of the mysql-package
```

```
const express = require('express'); // import of the express-
package/middleware
```

```
const port = 3000; // ThKind of a telefon number, on which the server
listens for requests
```

```
// Variable set with the db-connection credentials to connect the server to
the database
const config = {
  host: 'localhost',
  database: 'myDatabase',
  user: "restrictedUser",
  password: 'SafePassword123'
}
```

```
const connection = mysql.createConnection(config)
// connection line to the database, that takes the credentials
// and returns a open line to execute sql statements
```

```
// function that actually connects the server to the database
// result is successful or errormessage
connection.connect(function(err) {
  if (err) throw err;
  console.log('Connected to MySQL database:', connection.config.database);
```

```
   // Preparation of the sql statement, which will be send to the database
  var sqlstmt = 'SELECT sysdate()';
```

```
   connection.query(sqlstmt, function (err, result) {
       // the sqlstmt is send to the database
       if (err) throw err; // if everthing is fine, i receive the resultset
```

```
       console.table(result); // We can present the result as a table
       console.log(result);   // Or as a unformated string
   });
});
```

Vocabulary

English	Deutsch
scalable	erweiterbar

Volkan Demir

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/modul/m290/learningunits/lu08/theorie/02?rev=1730295187

Last update: **2024/10/30 14:33**