Pengembangan Aplikasi *Web*

Pertemuan Ke-7
(Pengenalan Node.js)

Noor Ifada

noor.ifada@trunojoyo.ac.id

S1 Teknik Informatika – Universitas Trunojoyo Madura (UTM)
Semester Gasal 2018-2019

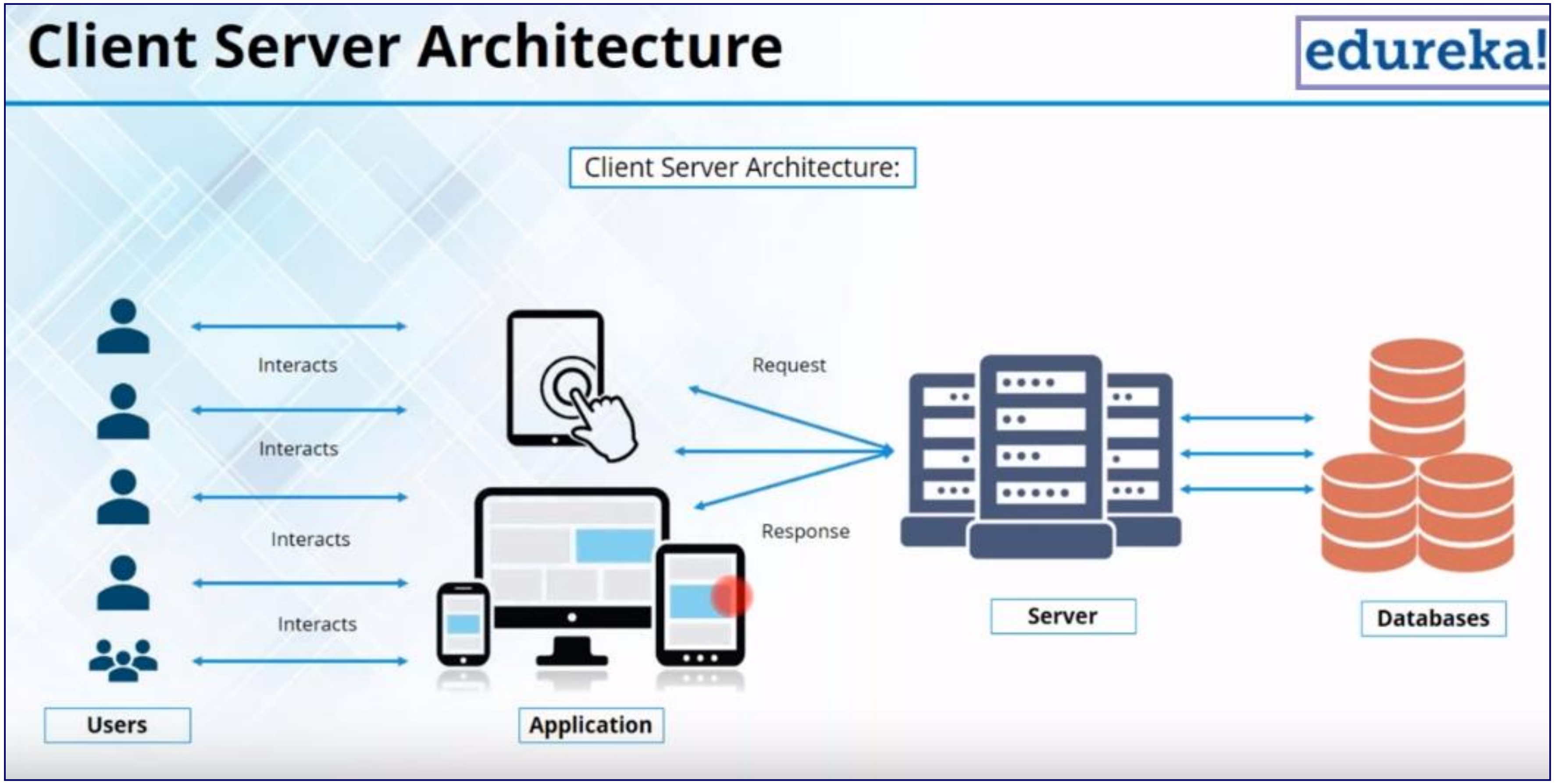
Sub Pokok Bahasan

- Pengenalan Node.js
- Instalasi *Software*
- Konsole Node.js
- *Module* dalam Node.js
- Node.js sebagai *Web Server*
- Node.js sebagai *File Server*
- NPM – *Node Package Manager*
- Node.js dan Sistem Basisdata

Pengenalan Node.js

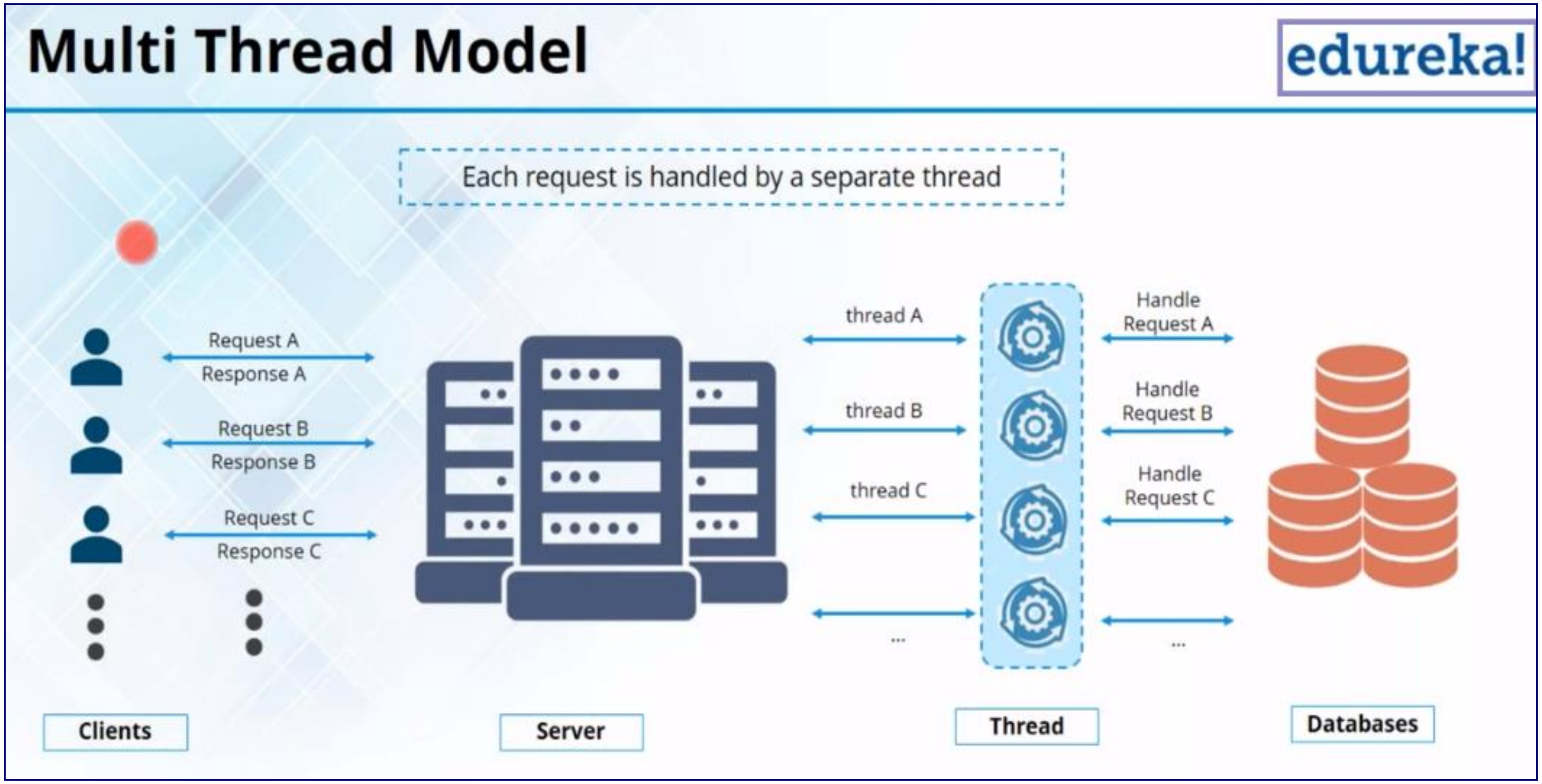
- *Open-source framework* dengan MIT *license* (https://nodejs.org)
- Menggunakan JavaScript untuk membangun aplikasi *server-side*
- Menggunakan *single-threaded model* dan bekerja secara *asynchronous* → bekerja lebih cepat daripada *framework* lain
- *Cross Platform*: Windows, MAC atau Linux

Arsitektur *Client-Server*



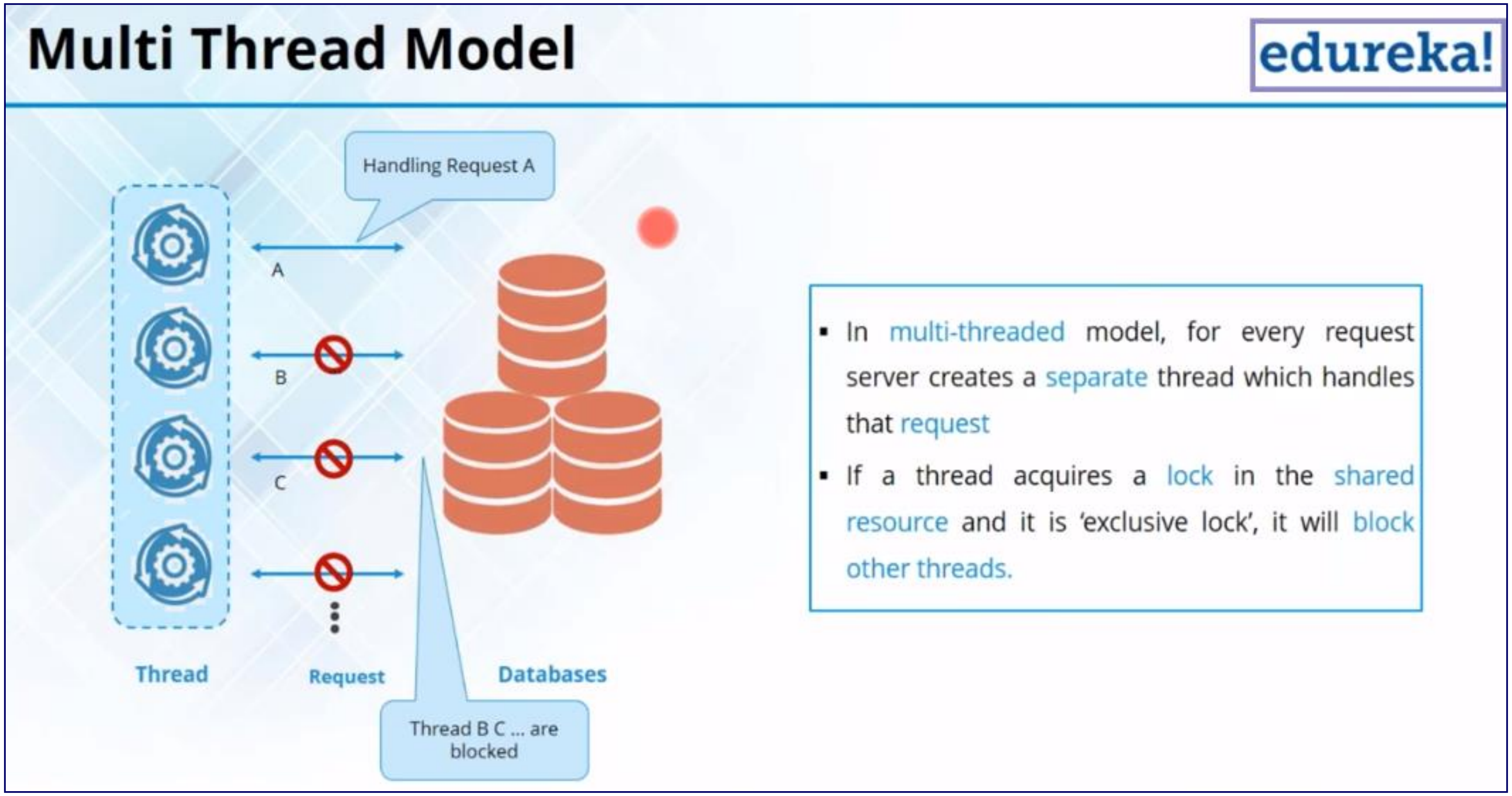
Source: https://www.youtube.com/watch?v=nZRbnBBupBI

Multi Thread Model



Source: https://www.youtube.com/watch?v=nZRbnBBupBI

Multi Thread Model [2]



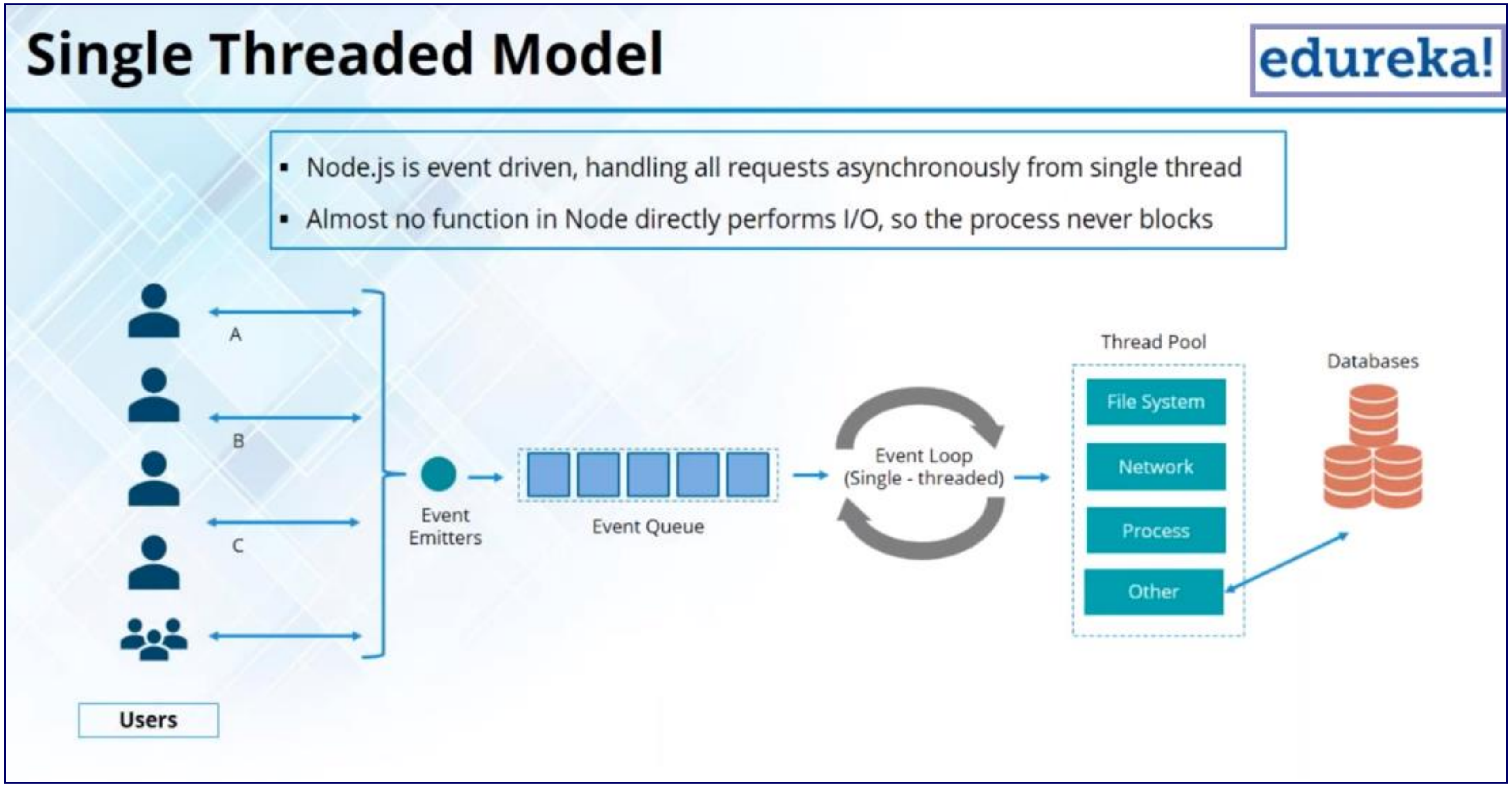
Multi Thread Model

edureka!

- In multi-threaded model, for every request server creates a separate thread which handles that request
- If a thread acquires a lock in the shared resource and it is 'exclusive lock', it will block other threads.

Source: https://www.youtube.com/watch?v=nZRbnBBupBI

Single Thread Model

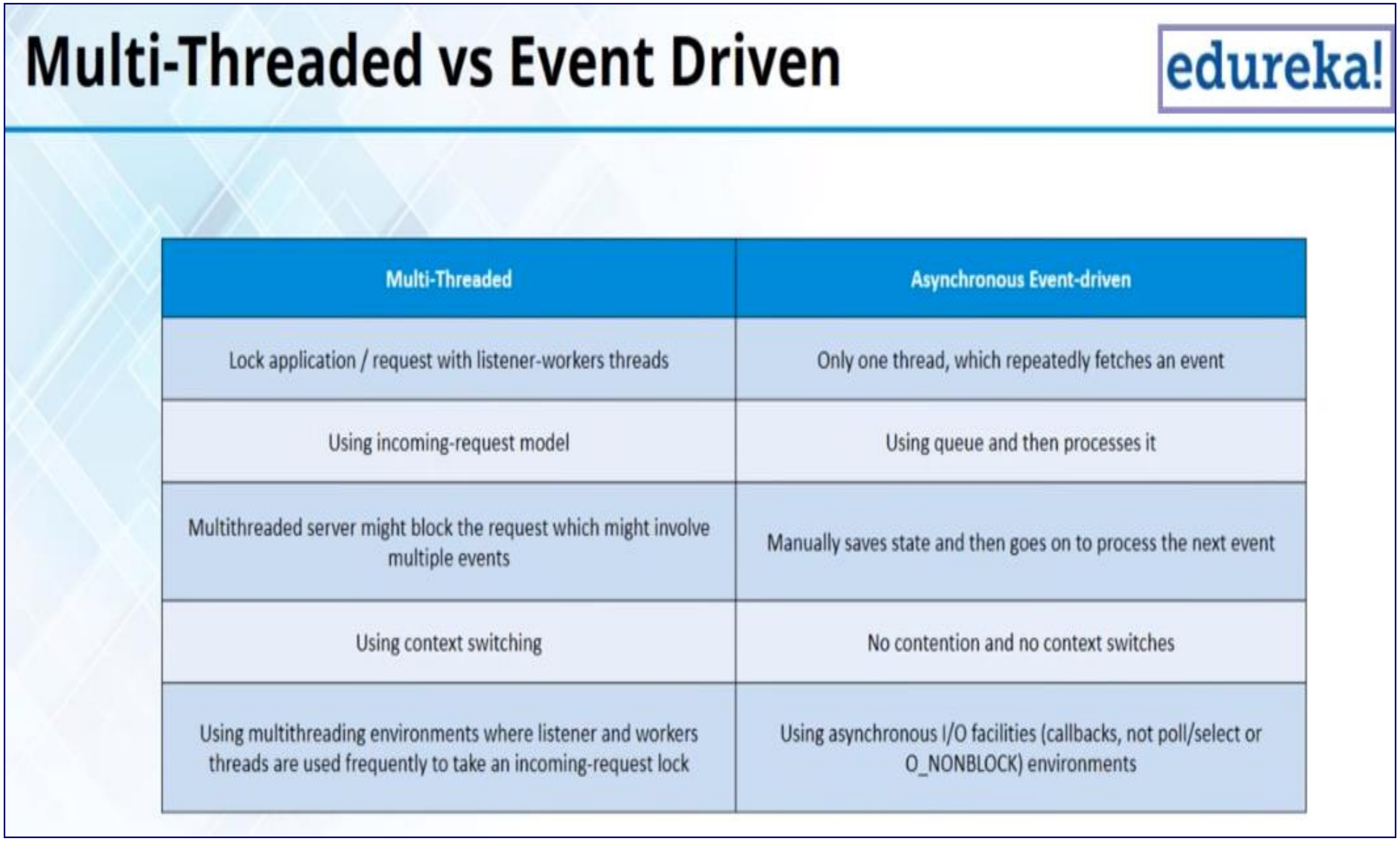


Single Threaded Model

- Node.js is event driven, handling all requests asynchronously from single thread
- Almost no function in Node directly performs I/O, so the process never blocks

Source: https://www.youtube.com/watch?v=nZRbnBBupBI

Multi-Thread VS Event Driven



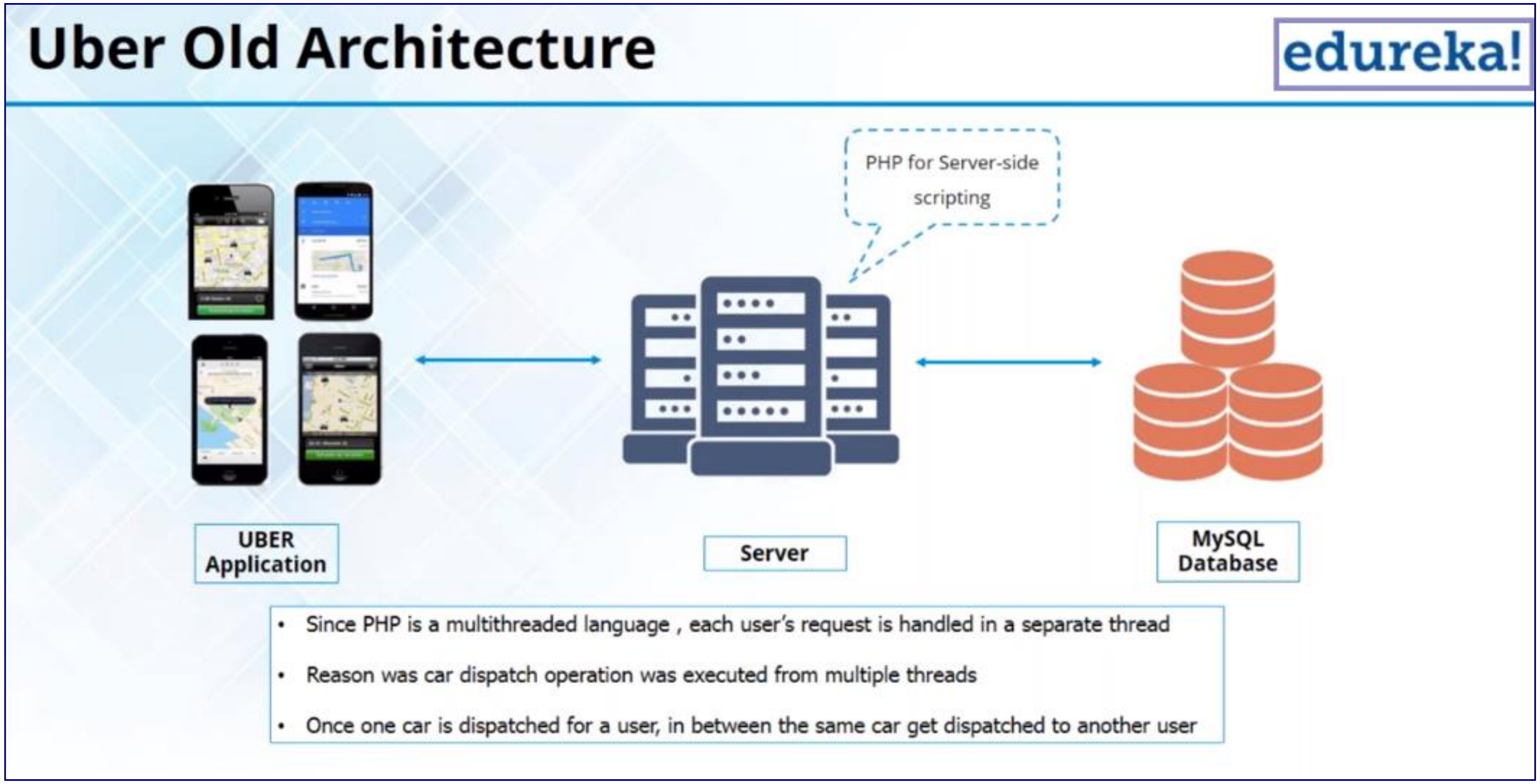
Multi-Threaded vs Event Driven

edureka!

Multi-Threaded	Asynchronous Event-driven
Lock application / request with listener-workers threads	Only one thread, which repeatedly fetches an event
Using incoming-request model	Using queue and then processes it
Multithreaded server might block the request which might involve multiple events	Manually saves state and then goes on to process the next event
Using context switching	No contention and no context switches
Using multithreading environments where listener and workers threads are used frequently to take an incoming-request lock	Using asynchronous I/O facilities (callbacks, not poll/select or O_NONBLOCK) environments

Source: https://www.youtube.com/watch?v=nZRbnBBupBI

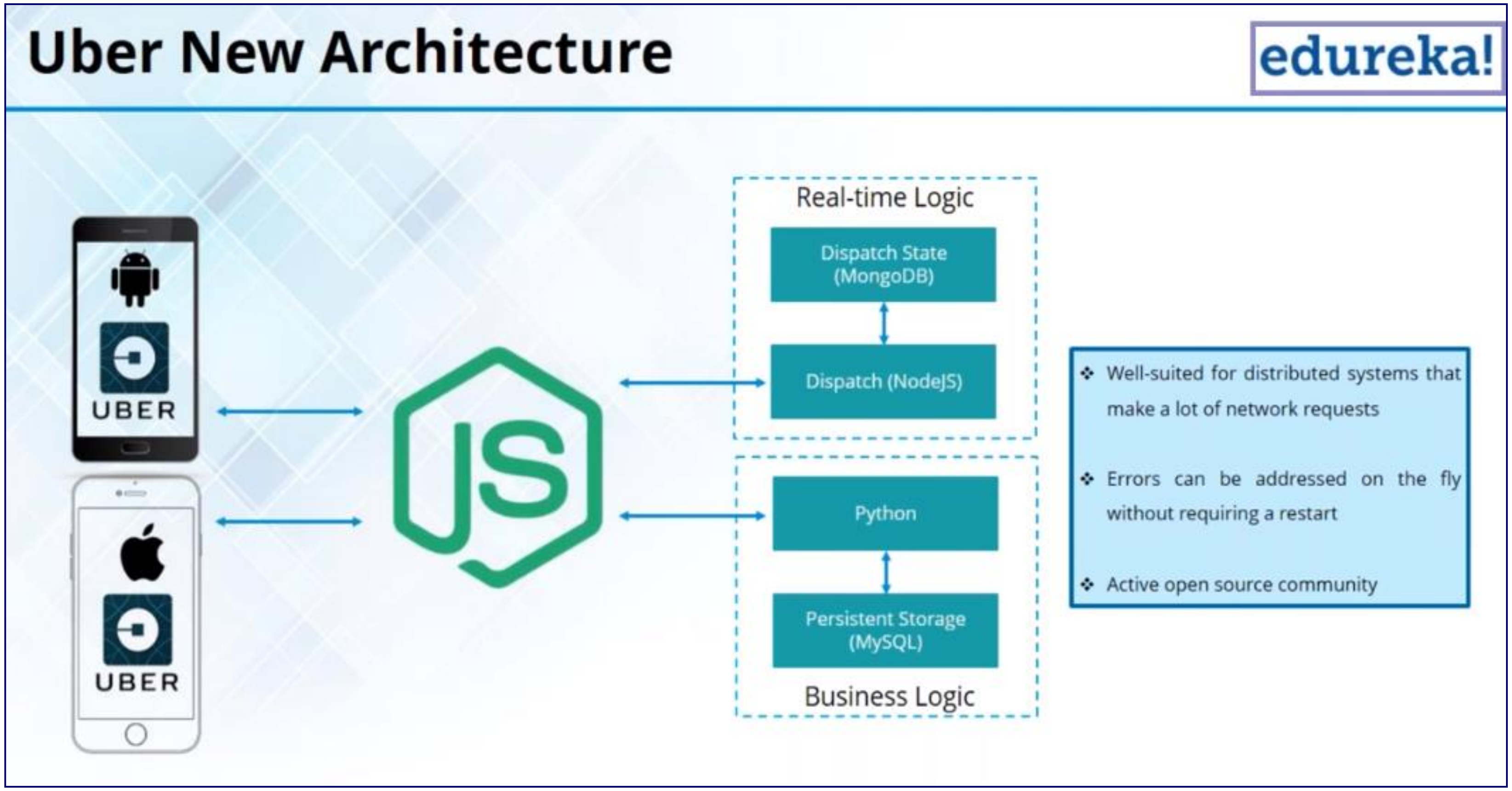
Contoh Aplikasi: Uber



- Since PHP is a multithreaded language , each user's request is handled in a separate thread
- Reason was car dispatch operation was executed from multiple threads
- Once one car is dispatched for a user, in between the same car get dispatched to another user

Source: https://www.youtube.com/watch?v=nZRbnBBupBI

Contoh Aplikasi: Uber [2]


Uber New Architecture
edureka!
UBER
UBER
JS
Real-time Logic
Dispatch State (MongoDB)
Dispatch (NodeJS)
Python
Persistent Storage (MySQL)
Business Logic
❖ Well-suited for distributed systems that make a lot of network requests
❖ Errors can be addressed on the fly without requiring a restart
❖ Active open source community

Source: https://www.youtube.com/watch?v=nZRbnBBupBI

Instalasi *Software*

Dibutuhkan	Digunakan dalam perkuliahan
Node.js	8.12.0 (includes npm 6.4.1)
Node Package Manager (NPM)	
Text Editor	Notepad++
Command Line Interface	Command Prompt
Web Browser	Chrome

NOTE: Cek dokumen Praktikum Bagian I!

Konsole Node.js

- Node.js memiliki *virtual environment* atau *Node shell* yang disebut sebagai REPL (*Read-Eval-Print-Loop*)
- Konsole digunakan untuk membuat dan menguji skrip Node.js/JavaScript code
- Sintaks JavaScript pada Node.js sama dengan sintaks JavaScript pada *web browser*

NOTE: Cek dokumen Praktikum Bagian II!

Module dalam Node.js

- ***Module*** Node.js ≈ JavaScript ***libraries***
- Merupakan sekumpulan fungsi/*function* yang dapat digunakan dalam aplikasi
- Macam-macam module:
  a) *Built-in*
  b) *User-defined*
- Cara menggunakan ***module***:

```
require('nama_module')
```

Module dalam Node.js: *Built-in* [2]

Module	Description
assert	Provides a set of assertion tests
buffer	To handle binary data
child_process	To run a child process
cluster	To split a single Node process into multiple processes
crypto	To handle OpenSSL cryptographic functions
dgram	Provides implementation of UDP datagram sockets
dns	To do DNS lookups and name resolution functions
domain	Deprecated. To handle unhandled errors
events	To handle events
fs	To handle the file system
http	To make Node.js act as an HTTP server
https	To make Node.js act as an HTTPS server.
net	To create servers and clients
os	Provides information about the operation system

Module dalam Node.js: *Built-in* [3]

Module	Description
path	To handle file paths
punycode	Deprecated. A character encoding scheme
querystring	To handle URL query strings
readline	To handle readable streams one line at the time
stream	To handle streaming data
string_decoder	To decode buffer objects into strings
timers	To execute a function after a given number of milliseconds
tls	To implement TLS and SSL protocols
tty	Provides classes used by a text terminal
url	To parse URL strings
util	To access utility functions
v8	
vm	To compile JavaScript code in a virtual machine
zlib	To compress or decompress files

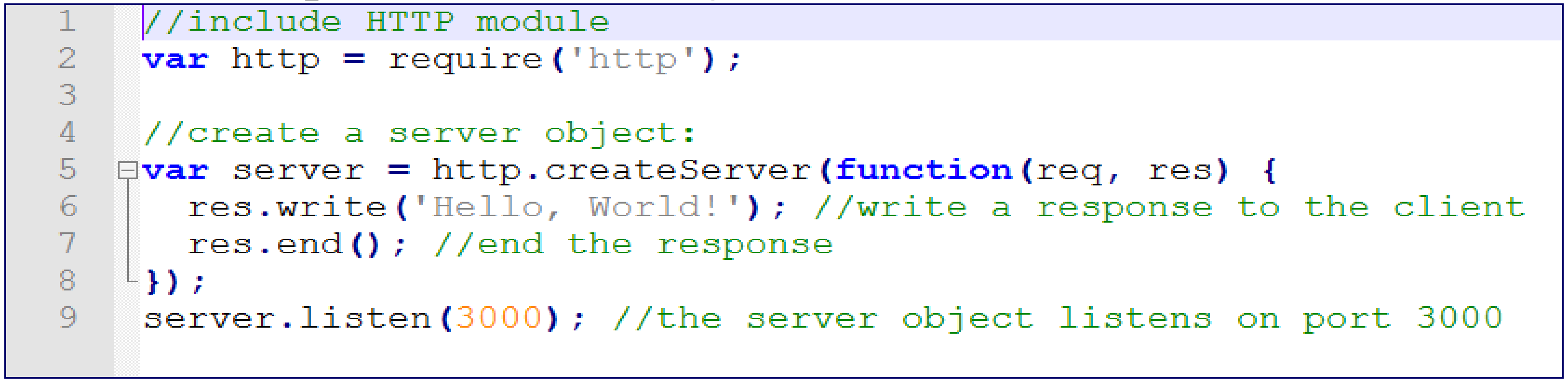
Source: https://www.w3schools.com/nodejs/ref_modules.asp

Node.js sebagai *Web Server*

- Gunakan ***module*** `http` agar Node.js dapat melakukan transfer data dengan menggunakan Hyper Text Transfer Protocol (HTTP)
- Gunakan ***method*** `createServer` untuk membuat HTTP *Server*
- Tambahkan HTTP header untuk dapat menampilkan *response* dari *web server* sesuai dengan tipe konten yang diinginkan

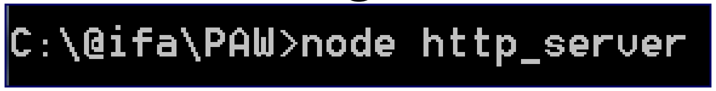
Node.js sebagai *Web Server* [2]

- Ketikan skrip berikut (nama *file*: **`http_server.js`**):



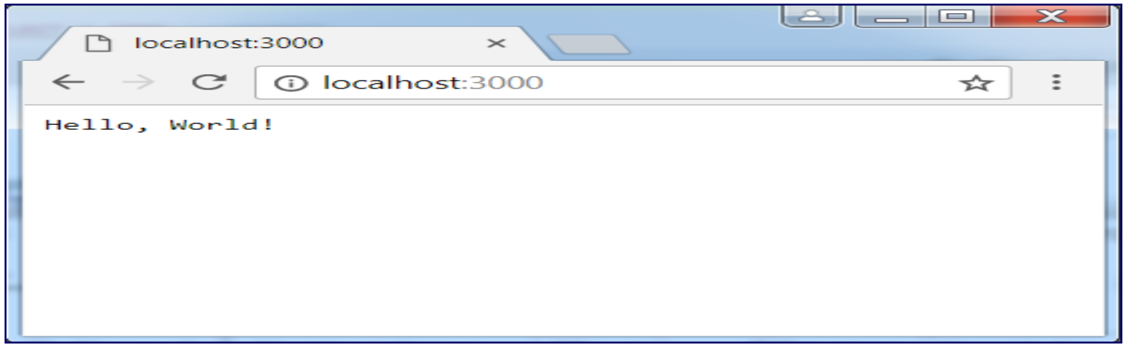
```
//include HTTP module
var http = require('http');

//create a server object:
var server = http.createServer(function(req, res) {
   res.write('Hello, World!'); //write a response to the client
   res.end(); //end the response
});
server.listen(3000); //the server object listens on port 3000
```

- Eksekusi program melalui ***command prompt***



```
C:\@ifa\PAW>node http_server
```

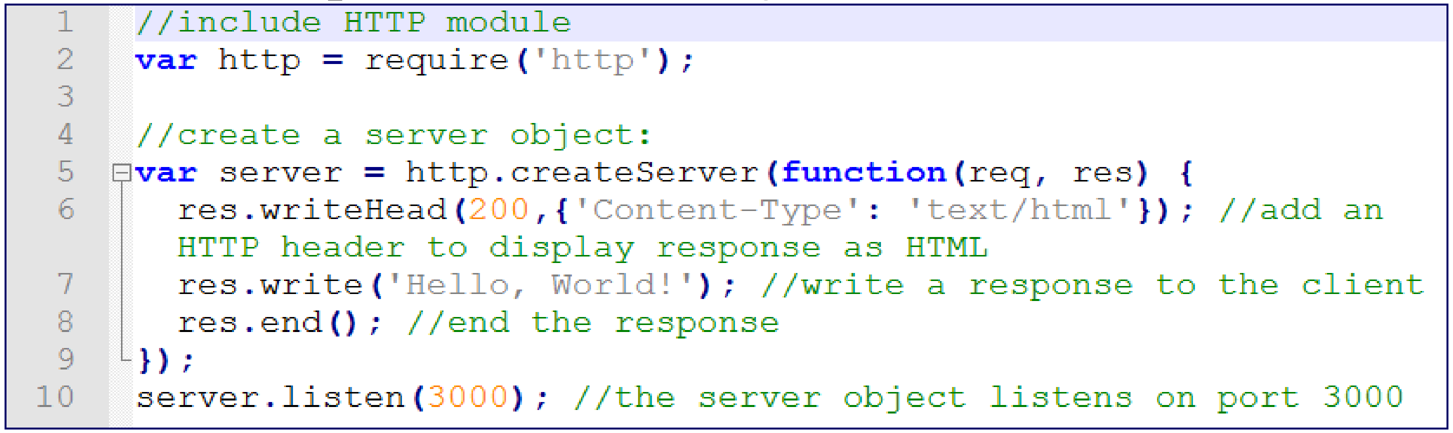
- Buka port 3000 melalui ***web browser***:



```
Hello, World!
```

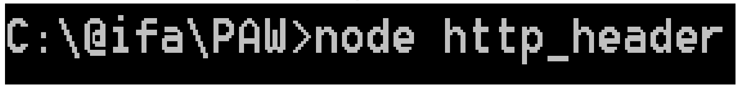
Node.js sebagai *Web Server* [3]

- Ketikan skrip berikut (nama *file*: **`http_header.js`**):



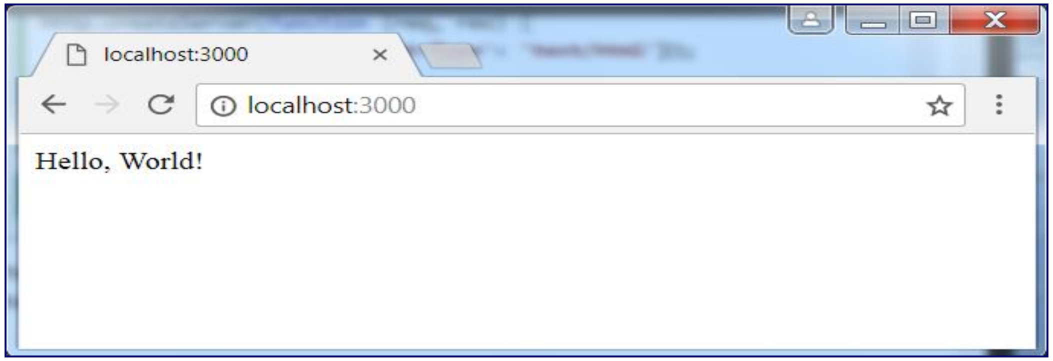
```
//include HTTP module
var http = require('http');

//create a server object:
var server = http.createServer(function(req, res) {
  res.writeHead(200,{'Content-Type': 'text/html'}); //add an HTTP header to display response as HTML
  res.write('Hello, World!'); //write a response to the client
  res.end(); //end the response
});
server.listen(3000); //the server object listens on port 3000
```

- Eksekusi program melalui ***command prompt***



```
C:\@ifa\PAW>node http_header
```

- Buka port 3000 melalui ***web browser***:



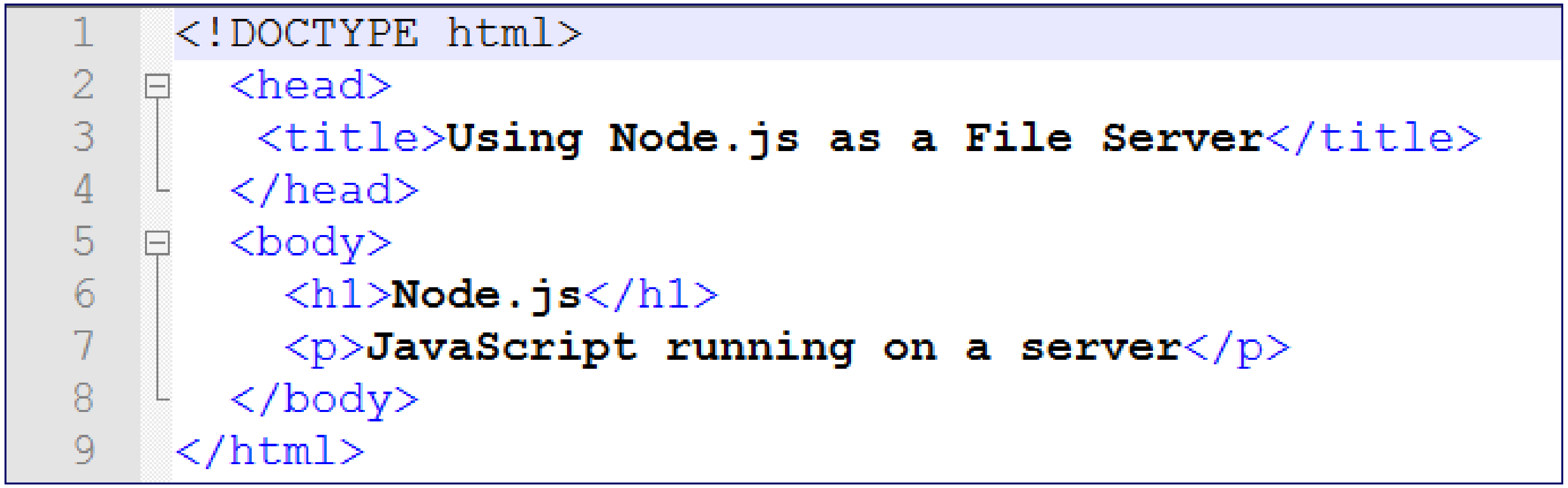
```
Hello, World!
```

Node.js sebagai *File Server*

- Gunakan ***module*** **`fs`** agar Node.js dapat berkerja dengan sistem *file* (*file system*) yang ada di komputer:
  - *Read files*
  - *Create files*
  - *Update files*
  - *Delete files*
  - *Rename files*

Node.js sebagai *File Server* [2]

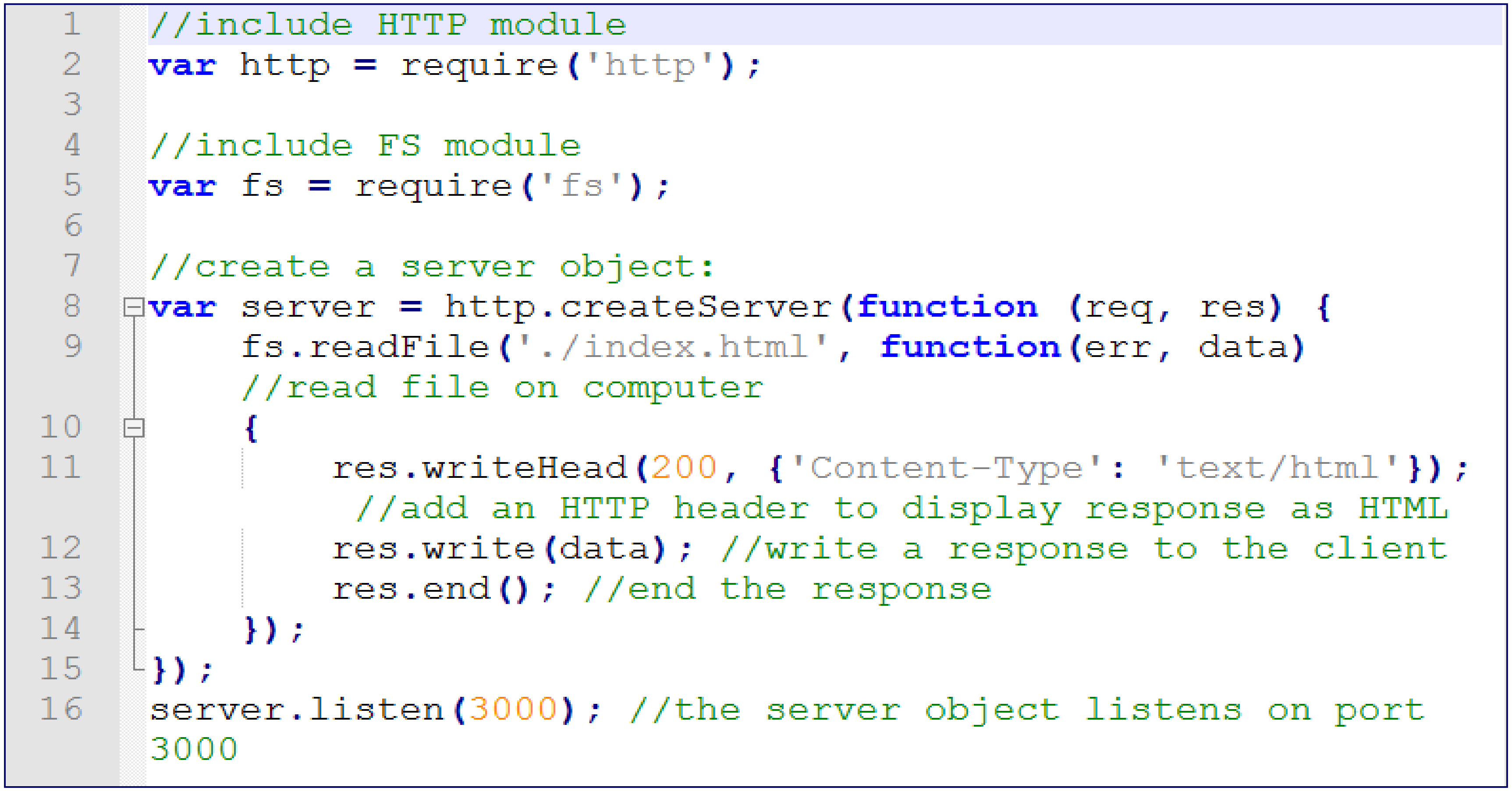
- Ketikan skrip berikut (nama *file*: **`index.html`**):



```
<!DOCTYPE html>
  <head>
   <title>Using Node.js as a File Server</title>
  </head>
  <body>
    <h1>Node.js</h1>
    <p>JavaScript running on a server</p>
  </body>
</html>
```

Node.js sebagai *File Server* [3]

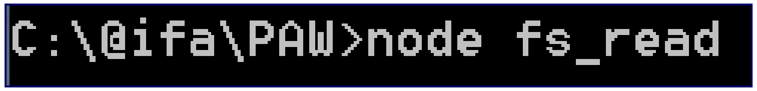
- Ketikan skrip berikut (nama *file*: **`fs_read.js`**):



```
//include HTTP module
var http = require('http');

//include FS module
var fs = require('fs');

//create a server object:
var server = http.createServer(function (req, res) {
    fs.readFile('./index.html', function(err, data) //read file on computer
    {
        res.writeHead(200, {'Content-Type': 'text/html'}); //add an HTTP header to display response as HTML
        res.write(data); //write a response to the client
        res.end(); //end the response
    });
});
server.listen(3000); //the server object listens on port 3000
```

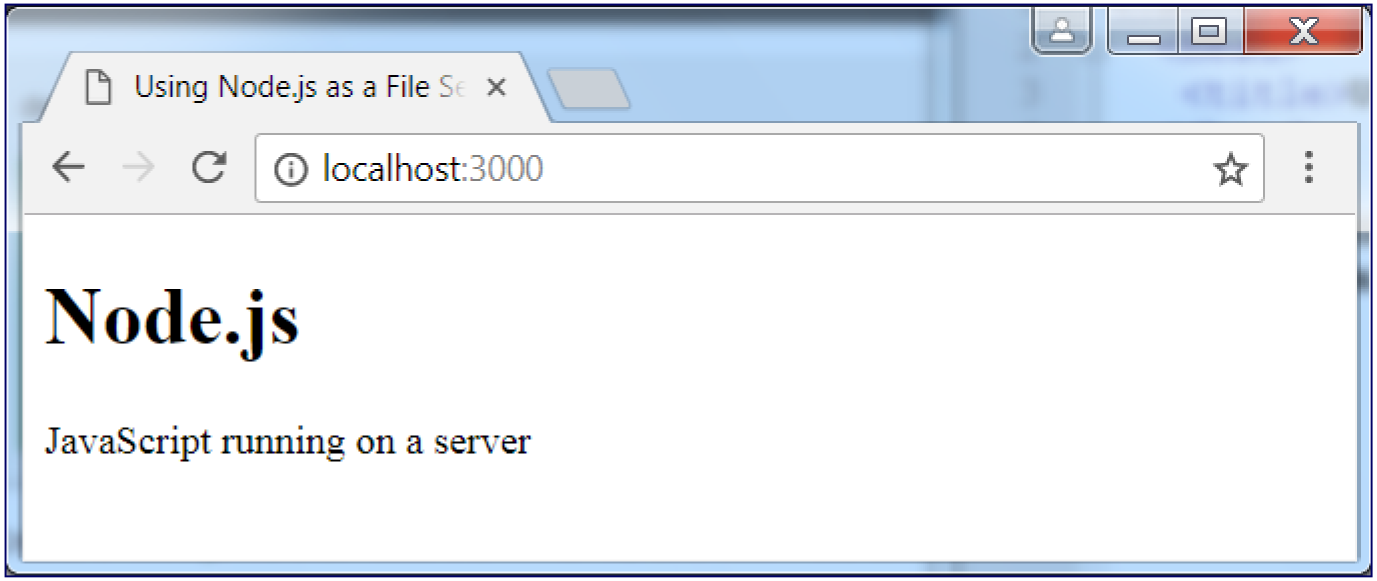
Node.js sebagai *File Server* [4]

- Eksekusi program melalui ***command prompt***



```
C:\@ifa\PAW>node fs_read
```

- Buka port 3000 melalui ***web browser***:



Node.js

JavaScript running on a server

NPM (Node.js *Package Manager*)

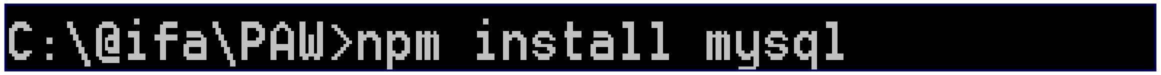
- Digunakan untuk melakukan instalasi module node
- Program **NPM** secara otomatis terinstal ketika Node.js terinstal
- Cara download ***package***:

```
npm install nama_package
```

- Daftar module yang terinstal ada di dalam *folder* "**node_modules**"

Node.js dan Sistem Basisdata

- Node.js dapat digunakan untuk mengakses sistem basis data
- *Download* dan lakukan instalasi module "mysql" lewat NPM agar basisdata MySQL dapat diakses dengan menggunakan Node.js:

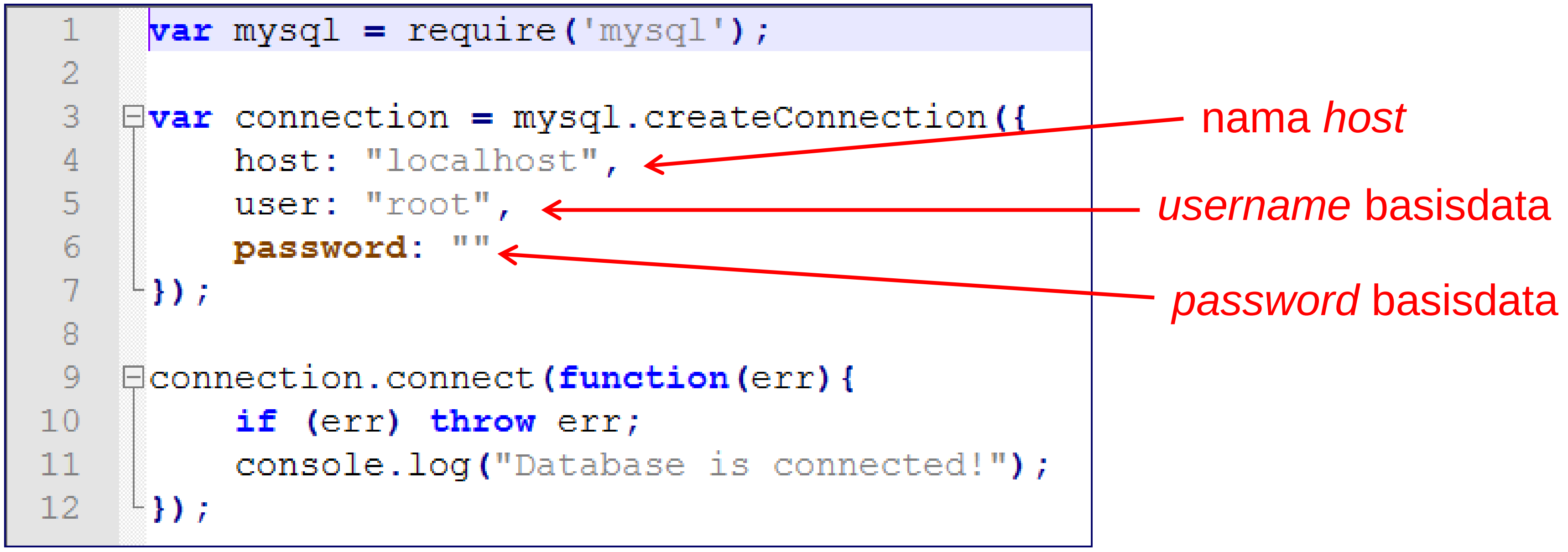


```
C:\@ifa\PAW>npm install mysql
```

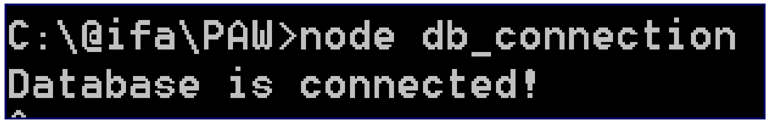
Membuat Koneksi ke Basisdata

- Untuk melakukan koneksi, gunakan nama *host*, *username* dan *password* yang digunakan untuk mengakses basisdata **(*NOTE*: pastikan server basisdata sudah diaktifkan)**

db_connection.js



```
var mysql = require('mysql');

var connection = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: ""
});

connection.connect(function(err){
    if (err) throw err;
    console.log("Database is connected!");
});
```

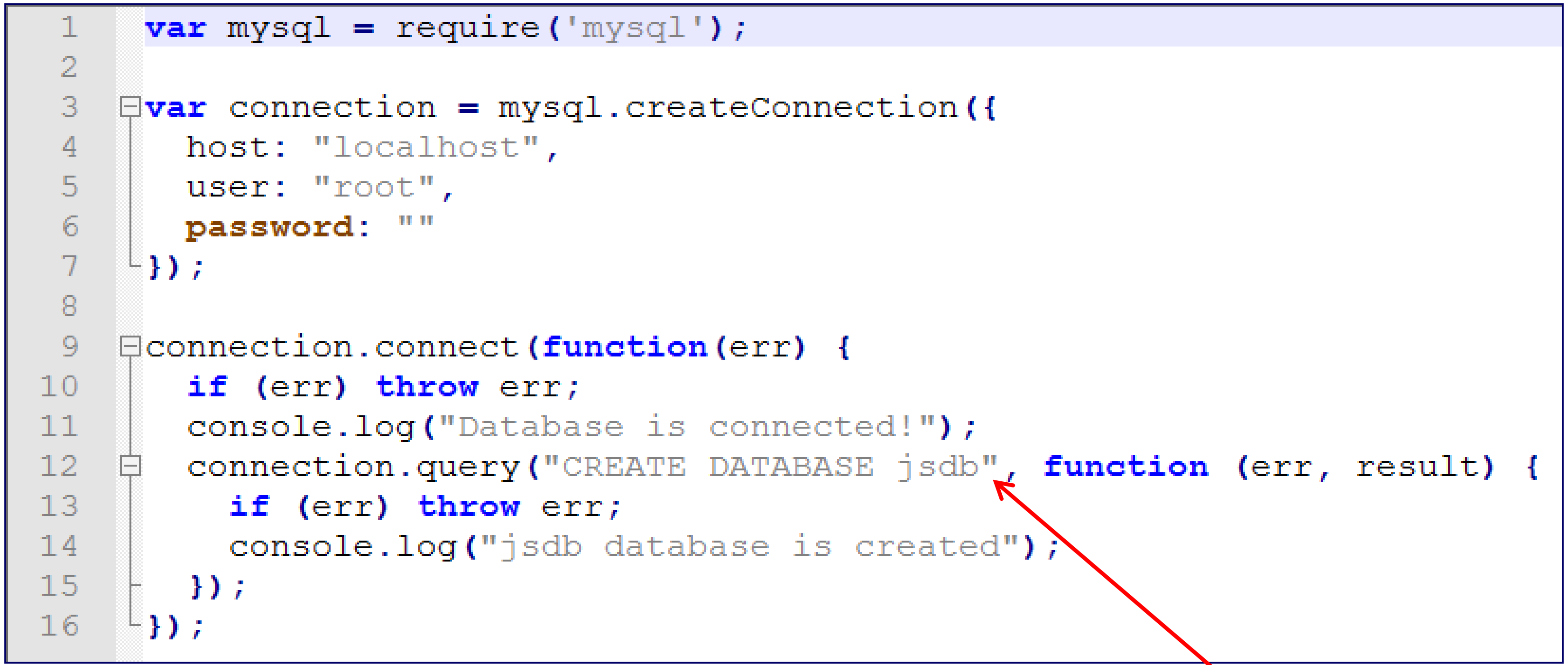


```
C:\@ifa\PAW>node db_connection
Database is connected!
```

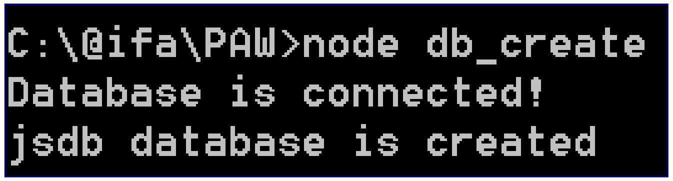
Query: Membuat (*Create*) Basisdata

- Gunakan perintah **CREATE DATABASE** untuk membuat basisdata baru

db_create.js



```
var mysql = require('mysql');

var connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: ""
});

connection.connect(function(err) {
  if (err) throw err;
  console.log("Database is connected!");
  connection.query("CREATE DATABASE jsdb", function (err, result) {
    if (err) throw err;
    console.log("jsdb database is created");
  });
});
```



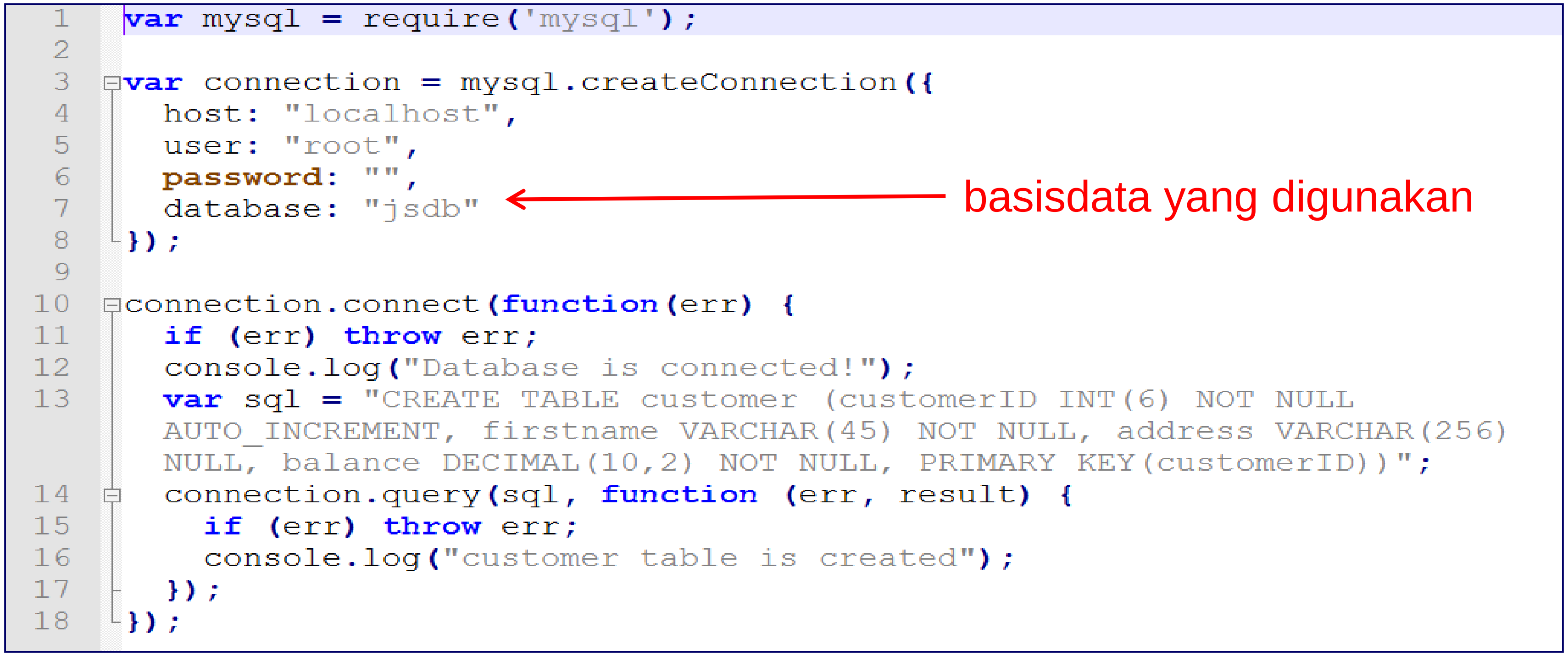
```
C:\@ifa\PAW>node db_create
Database is connected!
jsdb database is created
```

nama basisdata: **jsdb**

Query: Membuat (*Create*) Tabel

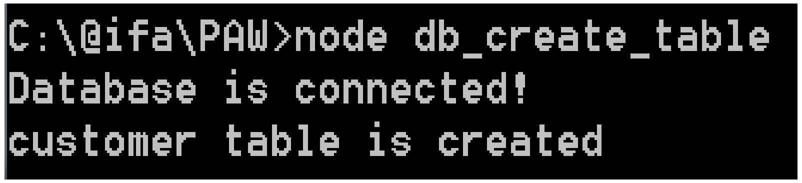
- Gunakan perintah **`CREATE TABLE`** untuk membuat tabel baru

db_create_table.js

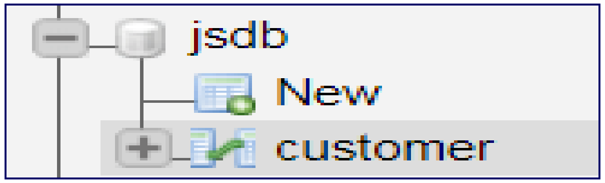


```
var mysql = require('mysql');

var connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "jsdb"
});

connection.connect(function(err) {
  if (err) throw err;
  console.log("Database is connected!");
  var sql = "CREATE TABLE customer (customerID INT(6) NOT NULL AUTO_INCREMENT, firstname VARCHAR(45) NOT NULL, address VARCHAR(256) NULL, balance DECIMAL(10,2) NOT NULL, PRIMARY KEY(customerID))";
  connection.query(sql, function (err, result) {
    if (err) throw err;
    console.log("customer table is created");
  });
});
```

basisdata yang digunakan



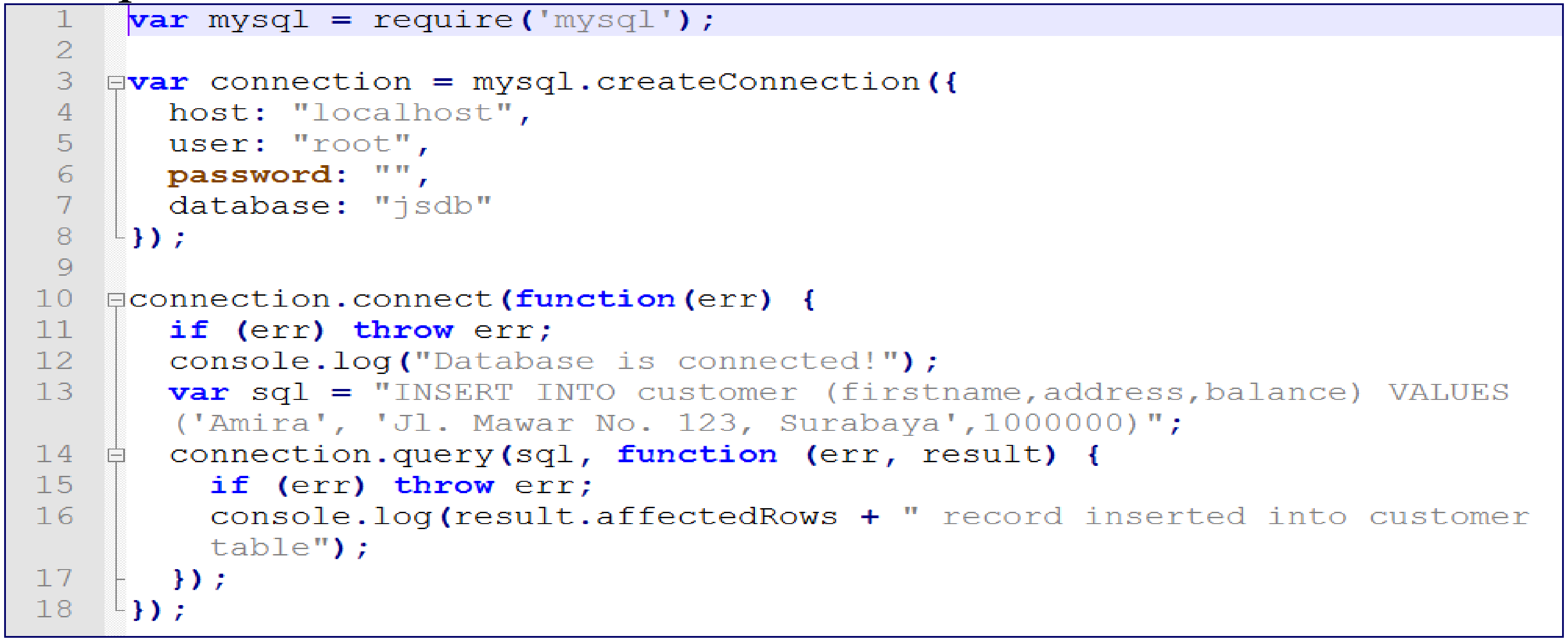
```
C:\@ifa\PAW>node db_create_table
Database is connected!
customer table is created
```



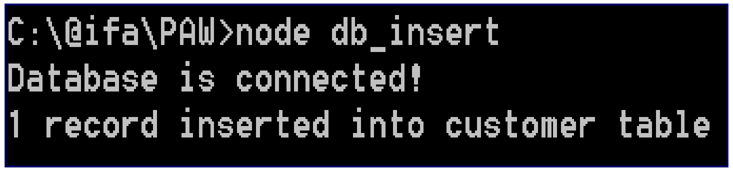
Query: Tambah Data (*Insert*) ke Tabel

- Gunakan perintah **`INSERT INTO`** untuk menambahkan data pada tabel

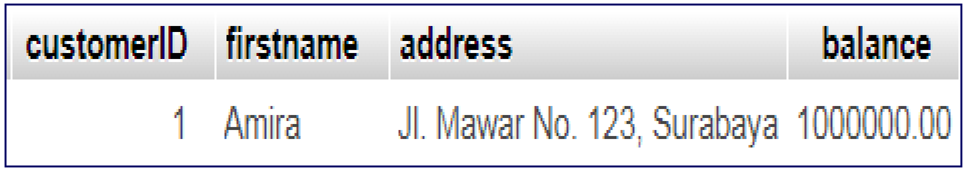
db_insert.js



```javascript
var mysql = require('mysql');

var connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "jsdb"
});

connection.connect(function(err) {
  if (err) throw err;
  console.log("Database is connected!");
  var sql = "INSERT INTO customer (firstname,address,balance) VALUES ('Amira', 'Jl. Mawar No. 123, Surabaya',1000000)";
  connection.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result.affectedRows + " record inserted into customer table");
  });
});
```



```
C:\@ifa\PAW>node db_insert
Database is connected!
1 record inserted into customer table
```

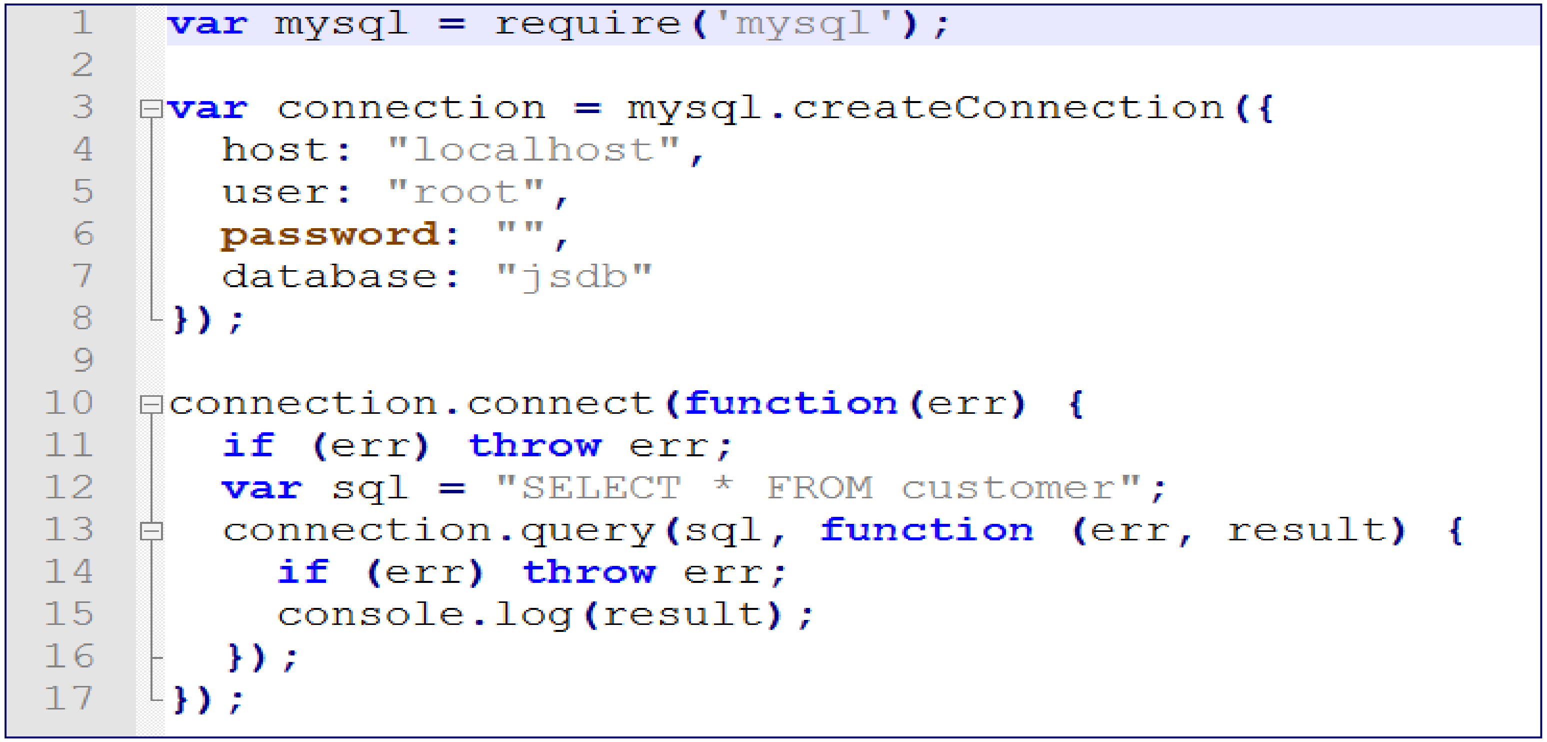


customerID	firstname	address	balance
1	Amira	Jl. Mawar No. 123, Surabaya	1000000.00

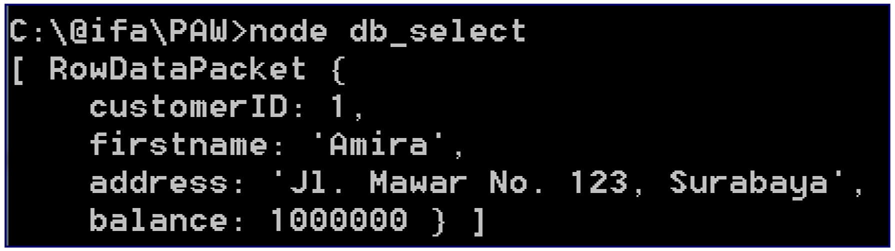
Query: Pilih Data (*Select*) dari Tabel

- Gunakan perintah **SELECT** untuk memilih data dari tabel

db_select.js



```javascript
var mysql = require('mysql');

var connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "jsdb"
});

connection.connect(function(err) {
  if (err) throw err;
  var sql = "SELECT * FROM customer";
  connection.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```

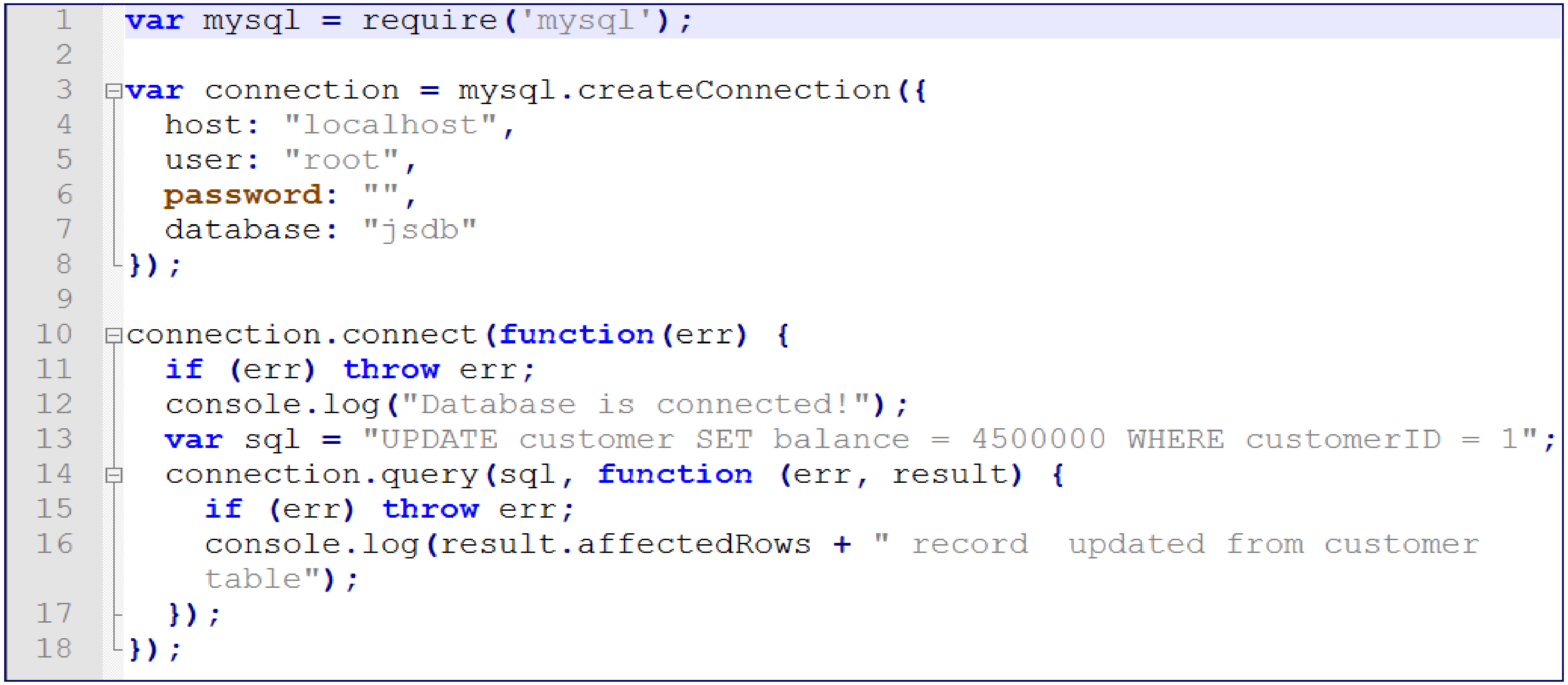


```
C:\@ifa\PAW>node db_select
[ RowDataPacket {
    customerID: 1,
    firstname: 'Amira',
    address: 'Jl. Mawar No. 123, Surabaya',
    balance: 1000000 } ]
```

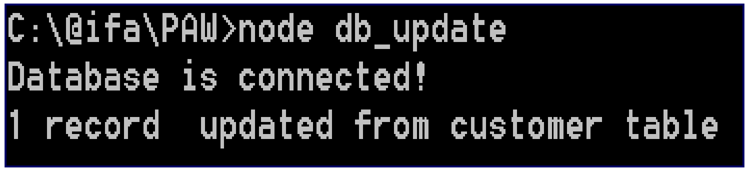
Query: Perbaharui Data (*Update*) Tabel

- Gunakan perintah **UPDATE** untuk memperbaharui data tabel

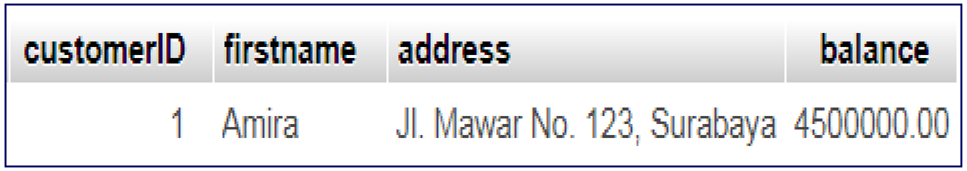
db_update.js



```
var mysql = require('mysql');

var connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "jsdb"
});

connection.connect(function(err) {
  if (err) throw err;
  console.log("Database is connected!");
  var sql = "UPDATE customer SET balance = 4500000 WHERE customerID = 1";
  connection.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result.affectedRows + " record  updated from customer table");
  });
});
```



```
C:\@ifa\PAW>node db_update
Database is connected!
1 record  updated from customer table
```

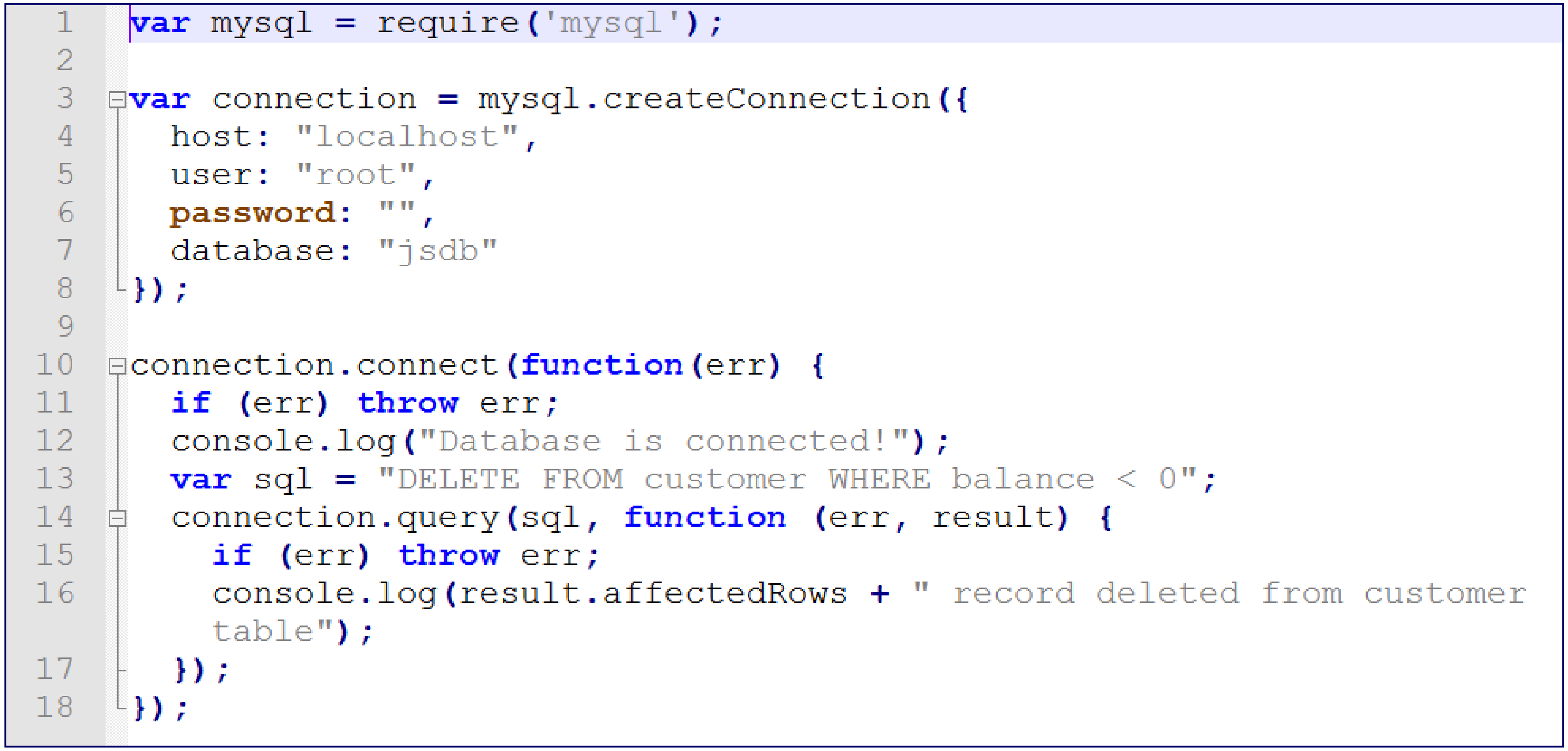


customerID	firstname	address	balance
1	Amira	Jl. Mawar No. 123, Surabaya	4500000.00

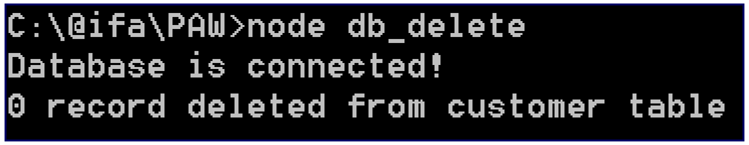
Query: Hapus Data (*Delete*) Tabel

- Gunakan perintah **DELETE** untuk menghapus data tabel

db_delete.js



```
var mysql = require('mysql');

var connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "jsdb"
});

connection.connect(function(err) {
  if (err) throw err;
  console.log("Database is connected!");
  var sql = "DELETE FROM customer WHERE balance < 0";
  connection.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result.affectedRows + " record deleted from customer table");
  });
});
```



```
C:\@ifa\PAW>node db_delete
Database is connected!
0 record deleted from customer table
```

Tugas

Ikuti sesi Praktikum 3 dan kerjakan Tugas Mingguan ke-4!