Node.js Example : Create a File

Following Node.js Example creates a file with data provided.

readFileExample.js

```
// include node fs module
var fs = require('fs');
var data ='Learn Node FS module';

// writeFile function with filename, content and callback function
fs.writeFile('newfile.txt', data, function (err) {
  if (err) throw err;
  console.log('File is created successfully.');
});
```

Run the program using node command in terminal or command prompt :

Terminal Output

```
$ node createFileExample.js
File is created successfully.
```

The file should be created next to your example node.js program with the content ‘Learn Node FS module’.

Node.js Example : Read a File

readFileExample.js

```
// include file system module
var fs = require('fs');

// read file sample.html
fs.readFile('sample.html',
    // callback function that is called when reading file is done
    function(err, data) {
        if (err) throw err;
        // data is a buffer containing file content
        console.log(data.toString('utf8'))
});
```

Run the program using node command in terminal or command prompt :

Terminal Output

```
$ node readFileExample.js
<html>
<body>
<h1>Header</h1>
<p>I have learnt to read a file in Node.js.</p>
</body>
</html>
```

Node.js Example : Delete a File

make sure there is a file named ‘sample.txt’ next to the node.js example program.

deleteFile.js

```
// include node fs module
var fs = require('fs');

// delete file named 'sample.txt'
fs.unlink('sample.txt', function (err) {
    if (err) throw err;
    // if no error, file has been deleted successfully
    console.log('File deleted!');
});
```

Run the program using node command in terminal or command prompt :

Terminal Output

```
$ node deleteFile.js
File deleted!
```

The file is successfully deleted.

Node.js Example : Write to a File

In this example, we shall write content, “Hello !” , to a text file sample.txt.

nodejs-write-to-file-example.js

```
// include file system module

var fs = require('fs');

var data = "Hello !"

// write data to file sample.html
fs.writeFile('sample.txt',data,
    // callback function that is called after writing file is done
    function(err) {
        if (err) throw err;
        // if no error
        console.log("Data is written to file successfully.")
});
```

When the above program is run in Terminal,

Program Output

```
arjun@arjun-VPCEH26EN:~/workspace/nodejs$ node nodejs-write-to-file-example.js
Data is written to file successfully.
```

NodeJS Example – Connect to MySQL Database

Npm install mysql

connectToMySQL.js - Connect to MySQL database in Node.js

```
// include mysql module
var mysql = require('mysql');

// create a connection variable with the details required
var con = mysql.createConnection({
  host: "localhost", // ip address of server running mysql
  user: "username", // user name to your mysql database
  password: "password" // corresponding password
});

// connect to the database.
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
});
```

```
$ node connectToMySQL.js
Connected!
```

NodeJS Example – SELECT FROM Table

selectFromTable.js Simple example to MySQL SELECT FROM query

```
// Node.js MySQL SELECT FROM query Example
// include mysql module
var mysql = require('mysql');

// create a connection variable with the required details
var con = mysql.createConnection({
  host: "localhost", // ip address of server running mysql
  user: "arjun", // user name to your mysql database
  password: "password", // corresponding password
  database: "studentsDB" // use the specified database
});

// make to connection to the database.
con.connect(function(err) {
  if (err) throw err;
  // if connection is successful
  con.query("SELECT * FROM students", function (err, result, fields) {
```

```
    // if any error while executing above query, throw error
    if (err) throw err;
    // if there is no error, you have the result
    console.log(result);
  });
});
```

```
$ node selectFromTable.js
[ RowDataPacket { name: 'John', rollno: 1, marks: 74 },
  RowDataPacket { name: 'Arjun', rollno: 2, marks: 74 },
  RowDataPacket { name: 'Prasanth', rollno: 3, marks: 77 },
  RowDataPacket { name: 'Adarsh', rollno: 4, marks: 78 },
  RowDataPacket { name: 'Raja', rollno: 5, marks: 94 },
  RowDataPacket { name: 'Sai', rollno: 6, marks: 84 },
  RowDataPacket { name: 'Ross', rollno: 7, marks: 54 },
  RowDataPacket { name: 'Monica', rollno: 8, marks: 86 },
  RowDataPacket { name: 'Lee', rollno: 9, marks: 98 },
  RowDataPacket { name: 'Bruce', rollno: 10, marks: 92 },
  RowDataPacket { name: 'Sukumar', rollno: 11, marks: 99 } ]
```

NodeJS Example – SELECT from Table with WHERE clause

We shall apply a filter based on marks and fetch only those records with marks greater than 90.

selectFromWhere.js

```
// include mysql module
var mysql = require('mysql');

// create a connection variable with the required details
var con = mysql.createConnection({
  host: "localhost", // ip address of server running mysql
  user: "arjun", // user name to your mysql database
  password: "password", // corresponding password
  database: "studentsDB" // use the specified database
});

// make to connection to the database.
con.connect(function(err) {
  if (err) throw err;
  // if connection is successful
  con.query("SELECT * FROM students where marks>90", function (err, result, fields) {
    // if any error while executing above query, throw error
    if (err) throw err;
    // if there is no error, you have the result
    console.log(result);
  });
});
```

Open a terminal from the location of above .js file and run selectFromWhere.js Node.js MySQL example program.

```
arjun@arjun-VPCEH26EN:~/workspace/nodejs$ node selectFromWhere.js
[ RowDataPacket { name: 'Raja', rollno: 5, marks: 94 },
  RowDataPacket { name: 'Lee', rollno: 9, marks: 98 },
  RowDataPacket { name: 'Bruce Wane', rollno: 10, marks: 92 },
  RowDataPacket { name: 'Sukumar', rollno: 11, marks: 99 } ]
```

NodeJS Example – ORDER entries BY a column

An example to sort entries in ascending order w.r.t a column.

AscOrderExample.js

```
// include mysql module
var mysql = require('mysql');

// create a connection variable with the required details
var con = mysql.createConnection({
  host: "localhost", // ip address of server running mysql
  user: "arjun", // user name to your mysql database
  password: "password", // corresponding password
  database: "studentsDB" // use the specified database
});
```

```
// make to connection to the database.
con.connect(function(err) {
  if (err) throw err;
  // if connection is successful
  con.query("SELECT * FROM students ORDER BY marks", function (err, result, fields) {
    // if any error while executing above query, throw error
    if (err) throw err;
    // if there is no error, you have the result
    console.log(result);
  });
});
```

Run the above Node.js MySQL ORDER BY example program.

```
arjun@arjun-VPCEH26EN:~/workspace/nodejs$ node AscOrderExample.js
[ RowDataPacket { name: 'Ross', rollno: 7, marks: 54 },
  RowDataPacket { name: 'John', rollno: 1, marks: 74 },
  RowDataPacket { name: 'Arjun', rollno: 2, marks: 74 },
  RowDataPacket { name: 'Prasanth', rollno: 3, marks: 77 },
  RowDataPacket { name: 'Adarsh', rollno: 4, marks: 78 },
  RowDataPacket { name: 'Sai', rollno: 6, marks: 84 },
  RowDataPacket { name: 'Monica Gellar', rollno: 8, marks: 86 },
  RowDataPacket { name: 'Bruce Wane', rollno: 10, marks: 92 },
  RowDataPacket { name: 'Raja', rollno: 5, marks: 94 },
```

```
  RowDataPacket { name: 'Lee', rollno: 9, marks: 98 },
  RowDataPacket { name: 'Sukumar', rollno: 11, marks: 99 } ]
```

The records are sorted in ascending order with respect to marks column.

NodeJS Example – INSERT entries INTO Table

```
// include mysql module
var mysql = require('mysql');

// create a connection variable with the required details
var con = mysql.createConnection({
  host: "localhost", // ip address of server running mysql
  user: "arjun", // user name to your mysql database
  password: "password", // corresponding password
  database: "studentsDB" // use the specified database
});

// make to connection to the database.
con.connect(function(err) {
  if (err) throw err;
  // if connection is successful
  con.query("INSERT INTO students (name,rollno,marks) values
('Anisha',12,95)", function (err, result, fields) {
```

```
    // if any error while executing above query, throw error
    if (err) throw err;
    // if there is no error, you have the result
    console.log(result);
  });
});
```

Run above Node.js MySQL program in Terminal.

```
arjun@arjun-VPCEH26EN:~/workspace/nodejs$ node InsertIntoExample.js
OkPacket {
  fieldCount: 0,
  affectedRows: 1,
  insertId: 0,
  serverStatus: 2,
  warningCount: 0,
  message: '',
  protocol41: true,
  changedRows: 0 }
```

Node.js Example – UPDATE Table Entries

UpdateRecordsFiltered.js - Update records of MySQL Table

```
// include mysql module
var mysql = require('mysql');

// create a connection variable with the required details
var con = mysql.createConnection({
  host: "localhost", // ip address of server running mysql
  user: "arjun", // user name to your mysql database
  password: "password", // corresponding password
  database: "studentsDB" // use the specified database
});

// make to connection to the database.
con.connect(function(err) {
  if (err) throw err;
  // if connection is successful
  con.query("UPDATE students SET marks=84 WHERE
marks=74", function (err, result, fields) {
    // if any error while executing above query, throw error
    if (err) throw err;
    // if there is no error, you have the result
    console.log(result);
  });
});
```

Run the above program in Terminal

Terminal Output

```
arjun@arjun-VPCEH26EN:~/workspace/nodejs$ node UpdateRecordsFiltered.js
OkPacket {
  fieldCount: 0,
  affectedRows: 3,
  insertId: 0,
  serverStatus: 34,
  warningCount: 0,
  message: '(Rows matched: 3  Changed: 3  Warnings: 0',
  protocol41: true,
  changedRows: 3 }
```

Node.js Example – DELETE Table Entries

Execute DELETE FROM query on specified table with filter applied on one or many properties of records in the table.

deleteRecordsFiltered.js

```
// include mysql module
var mysql = require('mysql');

// create a connection variable with the required details
var con = mysql.createConnection({
```

```
  host: "localhost", // ip address of server running mysql
  user: "arjun", // user name to your mysql database
  password: "password", // corresponding password
  database: "studentsDB" // use the specified database
});

// make connection to the database.
con.connect(function(err) {
  if (err) throw err;
  // if connection is successful
  con.query("DELETE FROM students WHERE rollno>10", function (err, result, fields) {
    // if any error while executing above query, throw error
    if (err) throw err;
    // if there is no error, you have the result
    console.log(result);
  });
});
```

Run deleteRecordsFiltered.js - Terminal Output

```
arjun@arjun-VPCEH26EN:~/workspace/nodejs$ node deleteRecordsFiltered.js
OkPacket {
  fieldCount: 0,
  affectedRows: 6,
  insertId: 0,
```

```
  serverStatus: 34,
  warningCount: 0,
  message: '',
  protocol41: true,
  changedRows: 0 }
```

Node.js Example – Using Result Object

We can access the records in Result Set as an array and properties of a record using DOT (.) Operator.

selectUseResultObject.js - Access rows and column data of result set

```
// Node.js MySQL Result Object Example
// include mysql module
var mysql = require('mysql');
// create a connection variable with the required details
var con = mysql.createConnection({
  host: "localhost", // ip address of server running mysql
  user: "arjun", // user name to your mysql database
  password: "password", // corresponding password
  database: "studentsDB" // use the specified database
});
// make to connection to the database.
con.connect(function(err) {
  if (err) throw err;
```

```
  // if connection is successful
  con.query("SELECT * FROM students", function (err, result, fields) {
    // if any error while executing above query, throw error
    if (err) throw err;
    // if there is no error, you have the result
    // iterate for all the rows in result
    Object.keys(result).forEach(function(key) {
      var row = result[key];
      console.log(row.name)
    });
  });
});
```

Run the above program using node in Terminal

Terminal Output

```
arjun@arjun-VPCEH26EN:~/workspace/nodejs$ node selectUseResultObject.js
John
Arjun
Prasanth
Adarsh
Raja
Sai
Ross
Monica
```

Lee
Bruce
Sukumar

Node.js Example – Parse URL Parameters

urlParsingExample.js - Node.js program to parse a URL into readable parts in Node.js

```
// include url module
var url = require('url');
var address = 'http://localhost:8080/index.php?type=page&action=update&id=5221';
var q = url.parse(address, true);

console.log(q.host); //returns 'localhost:8080'
console.log(q.pathname); //returns '/index.php'
console.log(q.search); //returns '?type=page&action=update&id=5221'

var qdata = q.query; // returns an object: { type: page, action: 'update',id='5221' }
console.log(qdata.type); //returns 'page'
console.log(qdata.action); //returns 'update'
console.log(qdata.id); //returns '5221'
```

Terminal Output

```
$ node urlParsingExample.js
localhost:8080
/index.php
?type=page&action=update&id=5221
page
update
5221
```

Node.js Example : Parse JSON File

Following example helps you to use JSON.parse() function and access the elements from JSON Object.

nodejs-parse-json.js

```
// json data
var jsonData = '{"persons":[{"name":"John","city":"New
York"},{"name":"Phil","city":"Ohio"}]}';

// parse json
var jsonParsed = JSON.parse(jsonData);

// access elements
console.log(jsonParsed.persons[0].name);
```

Terminal Output for running nodejs-parse-json.js

```
arjun@arjun-VPCEH26EN:~/workspace/nodejs$ node nodejs-parse-json.js
John
```

Node.js Example : Create HTTP Web Server

Node.js Example – A HTTP Web Server that prepares a response with HTTP header and a message.

```
// include http module in the file
var http = require('http');

// create a server
http.createServer(function (req, res) {
   // http header
   // 200 - is the OK message
   // to respond with html content, 'Content-Type' should be 'text/html'
   res.writeHead(200, {'Content-Type': 'text/html'});
   res.write('Node.js says hello!'); //write a response to the client
   res.end(); //end the response
}).listen(9000); //the server object listens on port 9000
```

Run the Server

$ node httpWebServer.js

Open a browser and hit the url, “http://127.0.0.1:9000/”, to trigger a request to our Web Server.