

A Beginners Guide to Scripting with Node.js and the node-oracledb Module

Contents

- The What & Why about Node.js
- Enter the node-oracledb module
- How to set all of this up?
- Examples of the node-oracledb in action
- Demo
- Where to go from here?

Some History

- Javascript created in 1995 by Brendan Eich
- In 2009 Ryan Dahl created Node.js, a javascript run-time using the Chrome V8 Browser, allowing javascript to be run outside a web browser.
- Today, like html & CSS, javascript is a popular core web technology.

Node.js Basics

Node.js is an open source, cross platform runtime engine where you can run server side javascript.

With Node.js built in modules, for example HTTP, you can use it to receive and respond to requests, you do not need a web server and can create a pure Node.js app.

Node.js uses built-in modules, but you can download many more modules that extend the functionality of Node.js

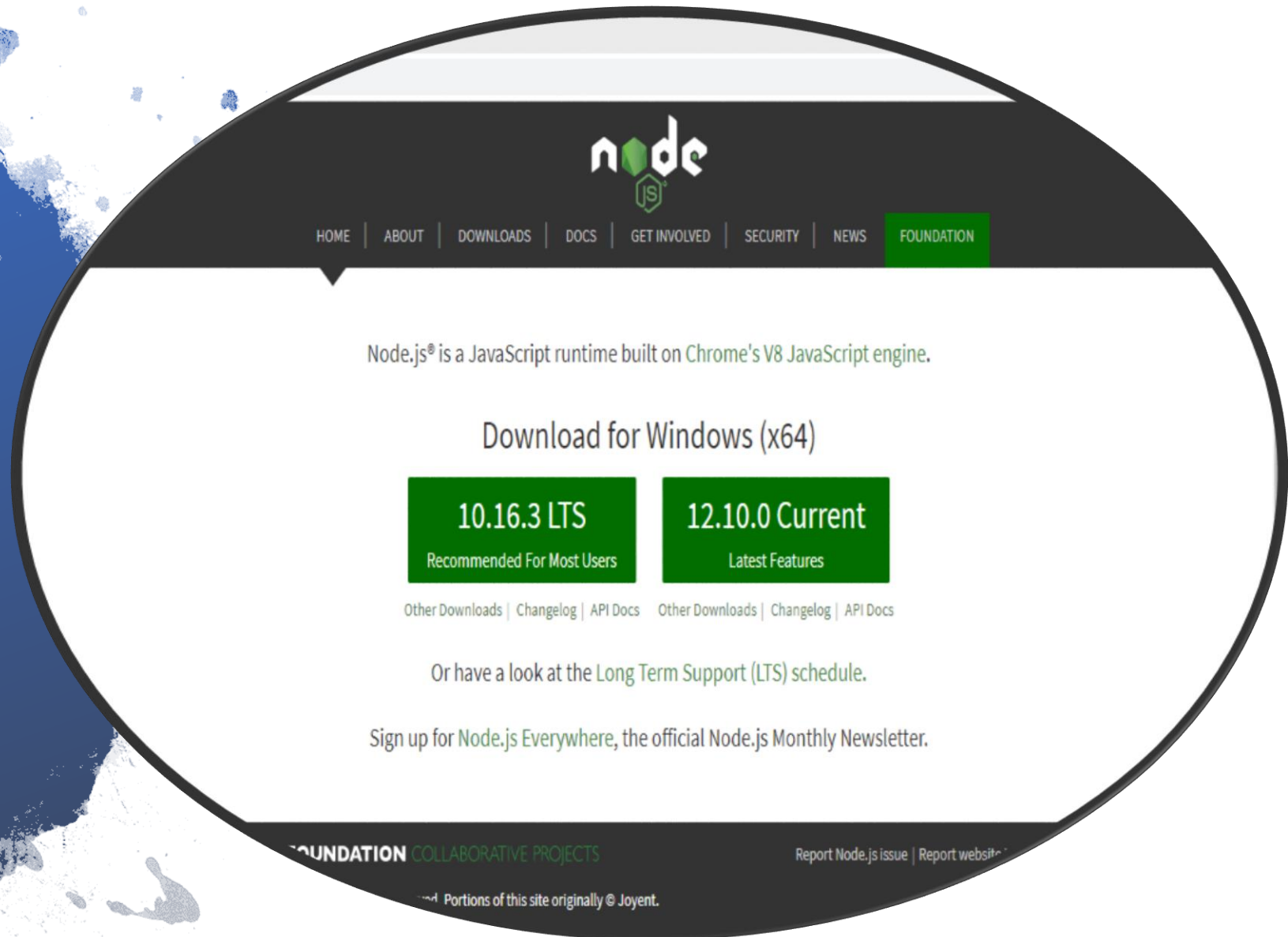
```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello, World!\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Setting up Node.js



Visit nodejs.org to download Node.js

Launch the Node.js command prompt to run npm, and launch node.js scripts



Launching Node.js & Running Scripts

```
Node.js command prompt - node test.js

C:\Users\Suzanne>node -v
v10.16.3

C:\Users\Suzanne>dir *.js
Volume in drive C is T110664900I
Volume Serial Number is CC1B-2A49

Directory of C:\Users\Suzanne

09/21/2019  05:52 PM                359 hellonyouworld.js
09/20/2019  04:38 PM                814 oracledb_connect_and_select.js
09/21/2019  05:57 PM                397 test.js
               3 File(s)            1,570 bytes
               0 Dir(s)  550,050,070,528 bytes free

C:\Users\Suzanne>node test.js
Server running at http://127.0.0.1:3000/
```

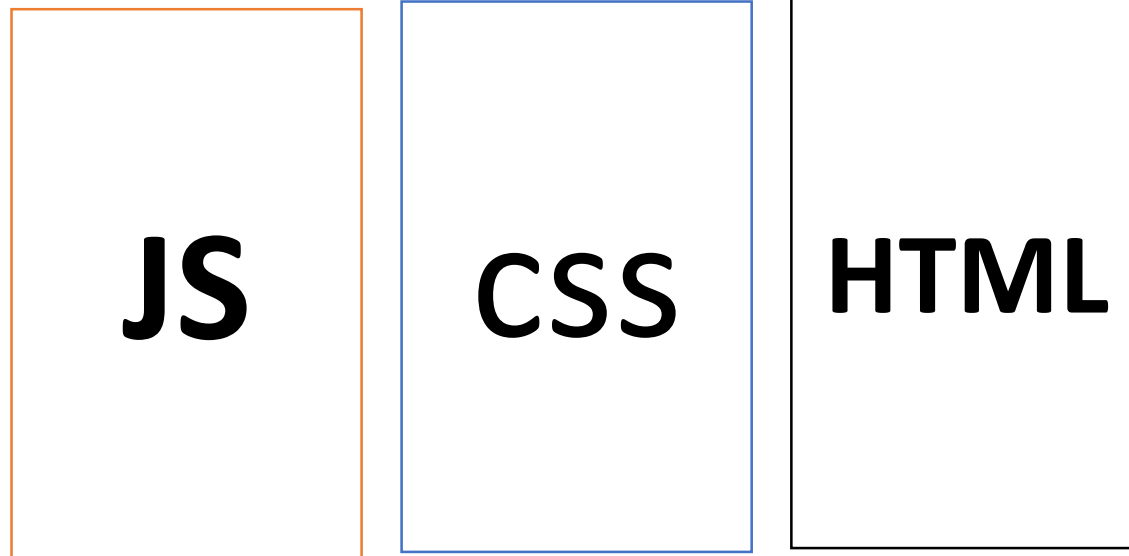
Run Node.js scripts by: node test.js

Why use Node.js & node-oracledb ?

You can take advantage of using the same language, Javascript on the front-end as well as the backend.

You can “modernize” shell scripts that are currently working with Oracle databases

Node-oracledb can talk to on-premise and cloud databases.



CORE WEB TECHNOLOGIES

Setting up node-oracledb

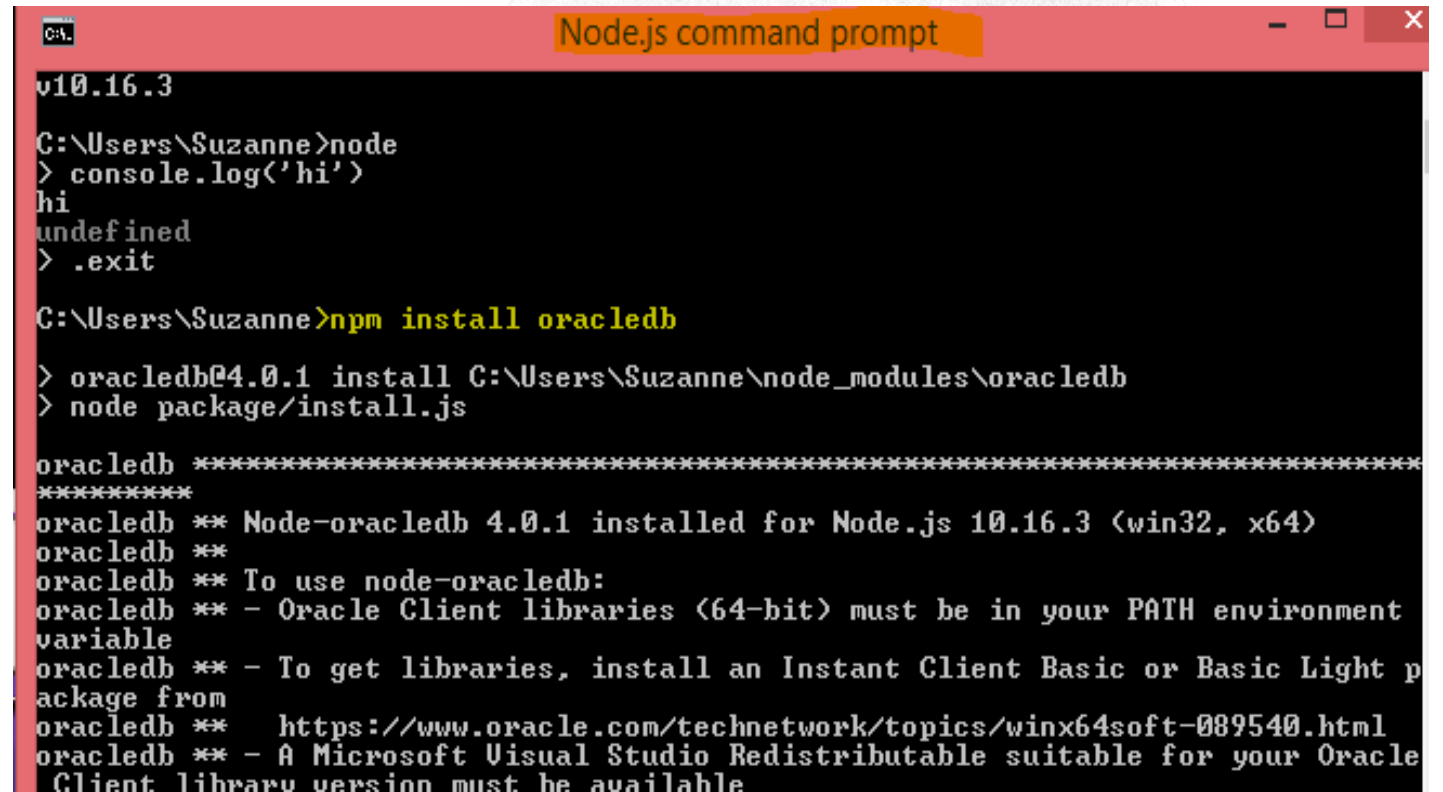
npm install -g oracledb

The node-oracledb module, is a high performance, open source module that you can download and use in Node.js scripts.

It is written and maintained by Oracle.

With it you can connect to your Oracle on premise and cloud databases.

Packages that are not built-in to Node.js can be downloaded through NPM.



```
C:\> Node.js command prompt
v10.16.3
C:\Users\Suzanne>node
> console.log('hi')
hi
undefined
> .exit

C:\Users\Suzanne>npm install oracledb
> oracledb@4.0.1 install C:\Users\Suzanne\node_modules\oracledb
> node package/install.js

oracledb *****
*****
oracledb ** Node-oracledb 4.0.1 installed for Node.js 10.16.3 (win32, x64)
oracledb **
oracledb ** To use node-oracledb:
oracledb ** - Oracle Client libraries (64-bit) must be in your PATH environment
variable
oracledb ** - To get libraries, install an Instant Client Basic or Basic Light p
ackage from
oracledb ** https://www.oracle.com/technetwork/topics/winx64soft-089540.html
oracledb ** - A Microsoft Visual Studio Redistributable suitable for your Oracle
Client library version must be available
```




Requirements for node-oracledb



node-oracledb version 4.0

[Documentation](#) [Installation](#) [Release Notes](#) [Source code](#) [Help](#)

About node-oracledb

The node-oracledb add-on for Node.js powers high performance Oracle Database applications.

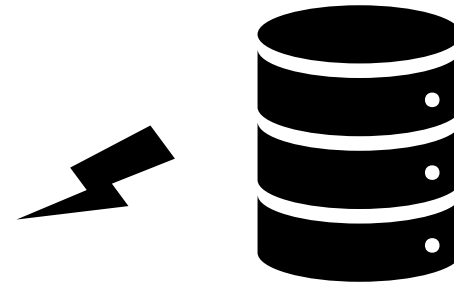
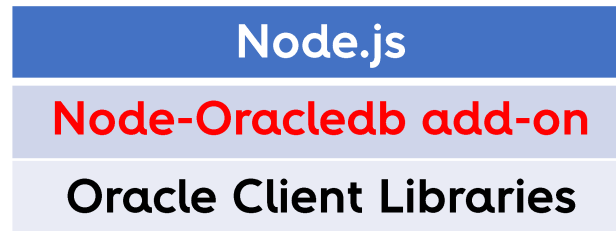
Use node-oracledb 4.0 to connect Node.js 8.16, 10.16, 12, or later, to Oracle Database. Older versions of node-oracledb may work with older versions of Node.js.

At the Node.js command window: Find out what version of node you are running to check compatibility with the node-oracledb module.

```
C:\Users\Suzanne>node -v  
v10.16.3
```

node-oracledb requires the Oracle Client

You must install the oracle client libraries

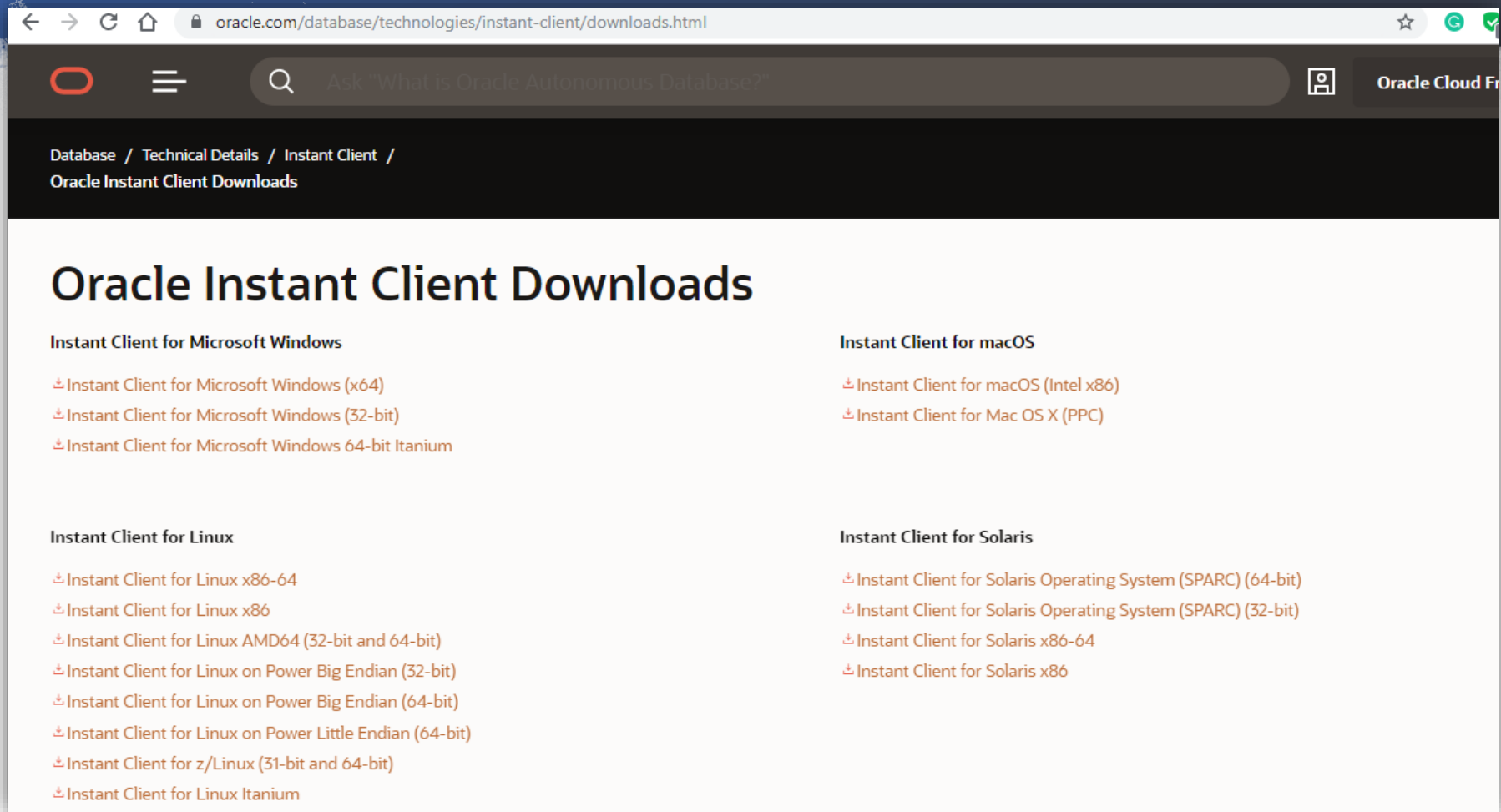


Oracle Database

<https://oracle.github.io/node-oracledb/INSTALL.html#quickstart>

Oracle Instant Client Downloads

<https://www.oracle.com/database/technologies/instant-client/downloads.html>



The screenshot shows a web browser window displaying the Oracle Instant Client Downloads page. The browser's address bar shows the URL: [oracle.com/database/technologies/instant-client/downloads.html](https://www.oracle.com/database/technologies/instant-client/downloads.html). The page features a dark navigation bar with the Oracle logo, a search bar containing the text "Ask 'What is Oracle Autonomous Database?'", and a user profile icon labeled "Oracle Cloud Fr". Below the navigation bar, a breadcrumb trail reads "Database / Technical Details / Instant Client / Oracle Instant Client Downloads". The main content area is titled "Oracle Instant Client Downloads" and is organized into four columns, each representing a different operating system: Microsoft Windows, macOS, Linux, and Solaris. Each column lists specific client versions with download icons.

Database / Technical Details / Instant Client / Oracle Instant Client Downloads

Oracle Instant Client Downloads

Instant Client for Microsoft Windows

- Instant Client for Microsoft Windows (x64)
- Instant Client for Microsoft Windows (32-bit)
- Instant Client for Microsoft Windows 64-bit Itanium

Instant Client for macOS

- Instant Client for macOS (Intel x86)
- Instant Client for Mac OS X (PPC)

Instant Client for Linux

- Instant Client for Linux x86-64
- Instant Client for Linux x86
- Instant Client for Linux AMD64 (32-bit and 64-bit)
- Instant Client for Linux on Power Big Endian (32-bit)
- Instant Client for Linux on Power Big Endian (64-bit)
- Instant Client for Linux on Power Little Endian (64-bit)
- Instant Client for z/Linux (31-bit and 64-bit)
- Instant Client for Linux Itanium

Instant Client for Solaris

- Instant Client for Solaris Operating System (SPARC) (64-bit)
- Instant Client for Solaris Operating System (SPARC) (32-bit)
- Instant Client for Solaris x86-64
- Instant Client for Solaris x86

Simple Example using node- oracledb

```
const oracledb = require('oracledb');

oracledb.outFormat = oracledb.OUT_FORMAT_OBJECT;

const mypw = 'PASSWORD'; // set mypw to the hr schema password

async function run() {

  let connection;

  try {
    connection = await oracledb.getConnection( {
      user      : "SYSTEM",
      password  : mypw,
      connectString : "localhost/DEMO"
    });

    const result = await connection.execute(
      `SELECT WINNER_AGE
       FROM wta_tennis_new
       `
    );
    console.log(result.rows);

  } catch (err) {
    console.error(err);
  } finally {
    if (connection) {
      try {
        await connection.close();
      }
    }
  }
}
```

```
    } catch (err) {
      console.error(err);
    }
  }
}
run();
```

require('oracledb') is used to load
The node-oracledb add-on

Execution of a
Simple Example
using node-
oracledb

```
C:\Users\Suzanne>node oracledb_connect_and_select.js  
[ < WINNER_AGE: 27.31553730319999 >,  
  < WINNER_AGE: 32.8268309377 >,  
  < WINNER_AGE: 26.3381245722 >,  
  < WINNER_AGE: 26.965092402499998 >,  
  < WINNER_AGE: 20.83778234089999 >,  
  < WINNER_AGE: 20.314852840499988 >,  
  < WINNER_AGE: 21.6646132786 >,  
  < WINNER_AGE: 27.173169062299998 >,  
  < WINNER_AGE: 22.7926078029 >,  
  < WINNER_AGE: 28.6269678303 >,  
  < WINNER_AGE: 19.88227241619999 > ],
```

Running DDL Commands

```
4 try {  
5     connection = await oracledb.getConnection( {  
6         user          : "SYSTEM",  
7         password      : mypw,  
8         connectString : "localhost/DEMO"  
9     });  
10  
11     console.log('Let us Create a table.')12  
13     await connection.execute(  
14         `CREATE TABLE hr_people3 (id NUMBER, first_name VARCHAR2(50), last_name VARCHAR2(50))`);
```

Running DML Commands

```
console.log('Insert Data Example.')
```

```
sqlstring = `INSERT INTO hr_people3 VALUES (:1, :2 , :3)`;
```

```
binds = [ [1, "John" ,"Smith"], [2, "Mike","Jones" ] ];
```

```
// For a complete list of options see the documentation.
```

```
options = {
```

```
  autoCommit: true,
```

```
  // batchSize: true, // continue processing even if there are data errors
```

```
  bindDefs: [
```

```
    { type: oracledb.NUMBER },
```

```
    { type: oracledb.STRING, maxSize: 50 },
```

```
    { type: oracledb.STRING, maxSize: 50 },
```

```
  ]
```

```
};
```

```
result = await connection.executeMany(sqlstring, binds, options);
```

```
console.log("Number of rows inserted:", result.rowsAffected);
```

```
C:\Users\Suzanne>node inserts.js  
Insert Data Example.  
Number of rows inserted: 2
```

```
Connected to:  
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production  
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing opt  
ions
```

```
SQL> select * from hr_people3 ;
```

	ID	FIRST_NAME
Smith	1	John
Jones	2	Mike

```
SQL>
```


Executing PL/SQL

```
try {
  connection = await oracledb.getConnection( {
    user      : "SYSTEM",
    password  : mypw,
    connectString : "localhost/DEMO"
  });

  console.log('.....')





  const result = await connection.execute(
    `select myfunction() from dual`
  );

  console.log(result.rows);
}
```

```
C:\Users\Suzanne>node myfunction
[ [ <'MYFUNCTION()' : 0 ] ]
```

```
Node.js command prompt - sqlplus SYSTEM/PASSWORD
Wrote file afiedt.buf
 1 create or replace function myfunction RETURN NUMBER IS
 2 BEGIN
 3 DECLARE
 4   x NUMBER :=0 ;
 5   BEGIN
 6     return x ;
 7   END ;
 8* END ;
SQL> /
Function created.
SQL>
```

Advanced Supported Features of oracle-nodedb

← → ↻ 🏠 oracle.com/database/technologies/appdev/nodejs.html ☆    | 

node-oracledb Features

- Install with standard npm infrastructure
- Open Source under the Apache 2 license
- Maintained by Oracle
- Hosted on GitHub
- Async/Await, Promises, Callbacks and Streams
- SQL and PL/SQL execution
- Oracle Database High Availability features
- Oracle Net features including encryption
- Array Fetches and Bulk loading features

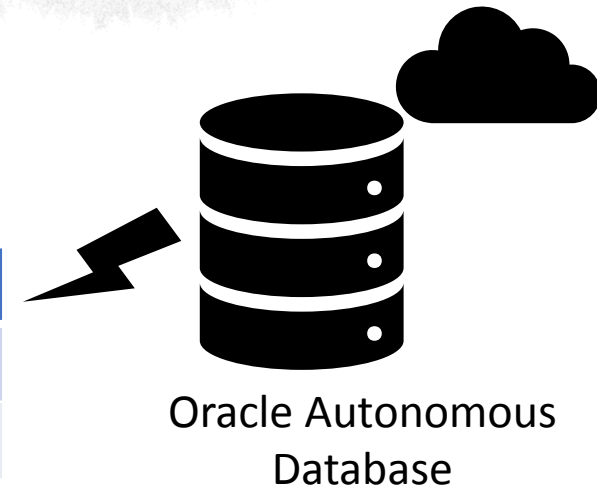
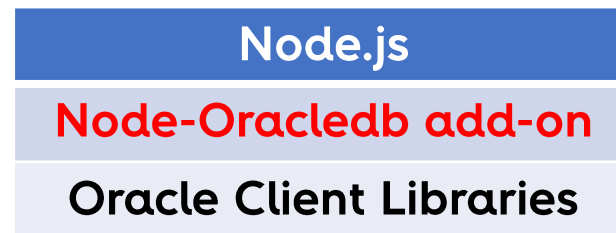
- Oracle Named Types and Collection support
- Large Objects: CLOBs and BLOBs as Streams or Strings and Buffers
- Oracle Database 12c JSON datatype
- Simple Oracle Document Access (SODA)
- Continuous Query Notification (CQN)
- Advanced Queuing (AQ)
- REF CURSORS and Implicit Results
- Data binding using JavaScript objects or arrays

- Inbuilt Connection Pool with Queuing, Aliasing, Tagging, Draining, Heterogeneous and Homogeneous connections, Proxy connections and Liveness checking
- Database Resident Connection Pooling (DRCP)
- Privileged connections
- External authentication
- Password changing
- Statement Saching and Client Result Caching
- End-to-end tracing, mid-tier authentication, and auditing

oracle-nodedb & Autonomous Cloud Connections

1. Get access to the Autonomous database
2. Install the Oracle Client
3. Download the credentials file
4. Use the oracle-nodedb add on

```
myfunction.js  
var oracledb = require('oracledb');  
  
oracledb.getConnection({  
  user: process.env.OADB_USER,  
  password: process.env.OADB_PW,  
  connectString: process.env.OADB_SERVICE  
},
```



Works Cited

- Source Code examples:
<https://oracle.github.io/node-oracledb/>
- <https://www.youtube.com/watch?v=zQtRwTOwisl>
Quickstart for Node.js and Oracle Database,
Christoper Jones, Oracle
- <https://github.com/oracle/node-oracledb/blob/master/examples/example.js#L32>
- <https://blogs.oracle.com/oraclemagazine/getting-started-with-autonomous>, Blaine Carter

Where to get More Information

- Oracle node-oracledb page:
<https://oracle.github.io/node-oracledb/>
- Node.js page:
<https://nodejs.org/en/about/>
- Javascript:
<https://developer.mozilla.org/en-US/docs/Web/JavaScript>