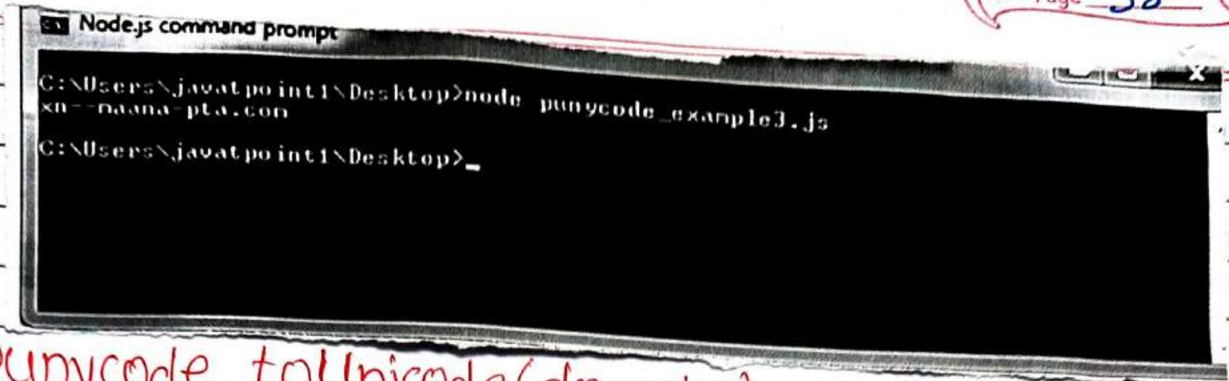


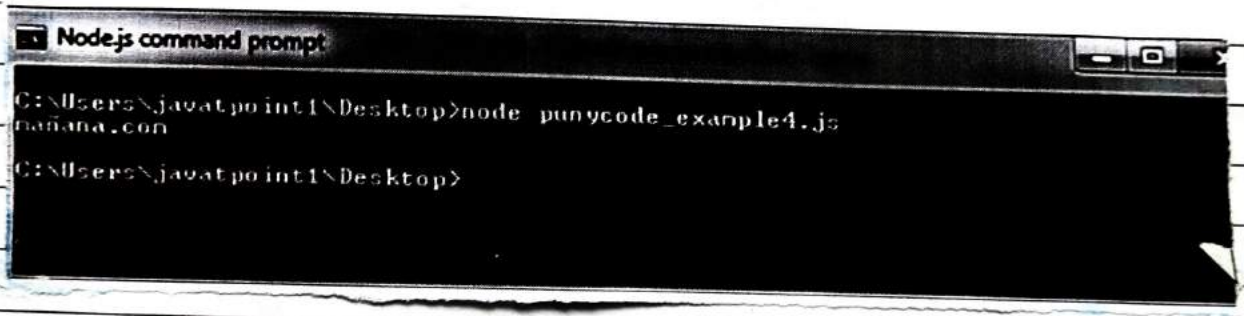
```
C:\Users\javatpoint1\Desktop>node punycode_example3.js
xn--maana-pta.com

C:\Users\javatpoint1\Desktop>_
```

Punycode. toUnicode(domain)

It is used to convert a Punycode string representing a domain name to Unicode. Only the Punycoded part of the domain name is converted.

File: punycode_example4.js

```
punycode = require('punycode');
console.log(punycode.toUnicode('xn-maana-pta.com'));
```



```
C:\Users\javatpoint1\Desktop>node punycode_example4.js
mañana.com

C:\Users\javatpoint1\Desktop>
```

Node.js TTY

The Node.js TTY module contains tty.ReadStream and tty.WriteStream classes. In most cases, there is no need to use this module directly.

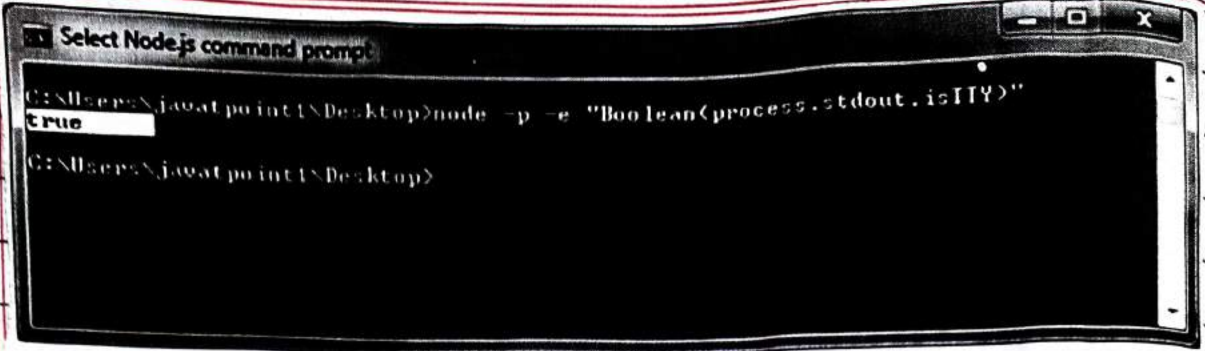
You have to use require('tty') to access this module

```
Var tty = require('tty')
```

When Node.js discovers that it is being run inside a TTY context, than:

- Process.stdin will be a tty.ReadStream instance.
- process.stdout will be a tty.WriteStream Instance.

To check that if Node.js is running in a TTY context, use the following command:



```
C:\Users\javatpoint1\Desktop>node -p -e "Boolean(process.stdout.isTTY)"
true

C:\Users\javatpoint1\Desktop>
```

Class: ReadStream

It contains a net.Socket subclass that represents the readable portion of a tty. In normal circumstances, the tty.ReadStream has the only instance named process.stdin in any Node.js program (Only When isatty(0)is true).

rs.isRaw:

It is a Boolean that it initialized to false. It specifi the current "raw" state of the tty.Readstream Instance.

rs.SetRawMode(mode)

It is should be true or false. It is used to set the properties of the tty.ReadStream to act either as a raw device or defalut, is Raw Will be set to the resulting mode.

Class: WriteStream

It contains a net.Socket subclass that represents the Writable portion.of a tty. In normal circumstances, the tty.WriteStream has the only instane named process.stdout it any Node.js program (only When isatty(1)is true).

Resize event: This event is used When either of the columns or raws properties has changed.

```
process.stdout.on('resize', () => {
  console.log('screen size has changed!');
  console.log('${process.stdout.columns} x ${process.
                 stdout.rows}');
});
```

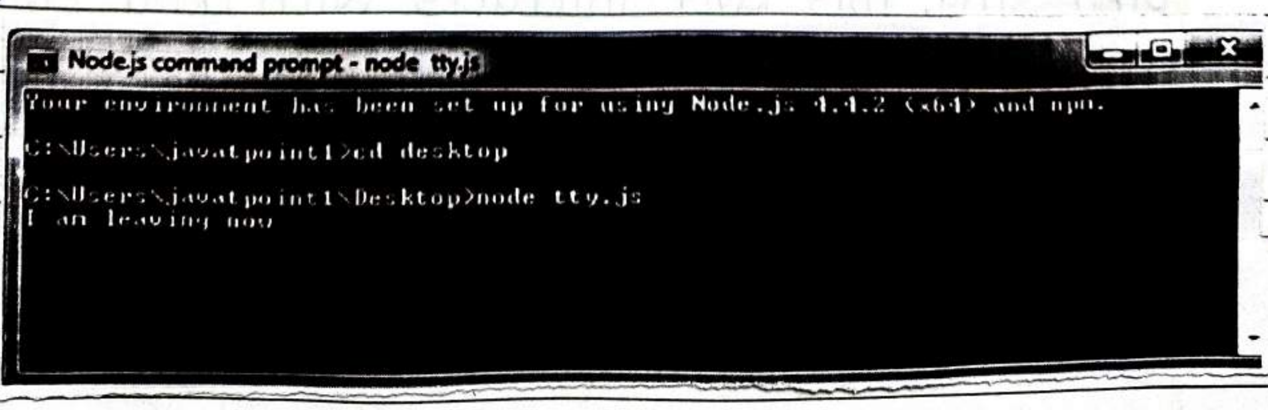
ws.columns:
It is used to give the number of columns the TTY currently has. This property gets updated on 'resize' events.

ws.rows:
It is used to give the number of rows the TTY currently has. This property gets updated on 'resize' events.

Node.js TTY Example

File: tty.js

```
var tty = require('tty');
process.stdin.setRawMode(true);
process.stdin.resume();
 console.log('I am leaving now');
process.stdin.on('keypress', function (char, key) {
  if (key && key.ctrl && key.name == 'c') {
   process.exit()
  }
});
```


Node.js command prompt - node tty.js

```
Your environment has been set up for using Node.js 4.4.2 (x64) and npm.

C:\Users\javatpoint1>cd desktop

C:\Users\javatpoint1\Desktop>node tty.js
I am leaving now
```

Node.js Web Module

What is Web Server

Web Server is a software program that handles HTTTP request sent by HTTP clients like Web browsers, and returns web pages in response to the clients. Web Servers usually respond with ht documents along with images, style sheets and scripts.

Web Application Architecture

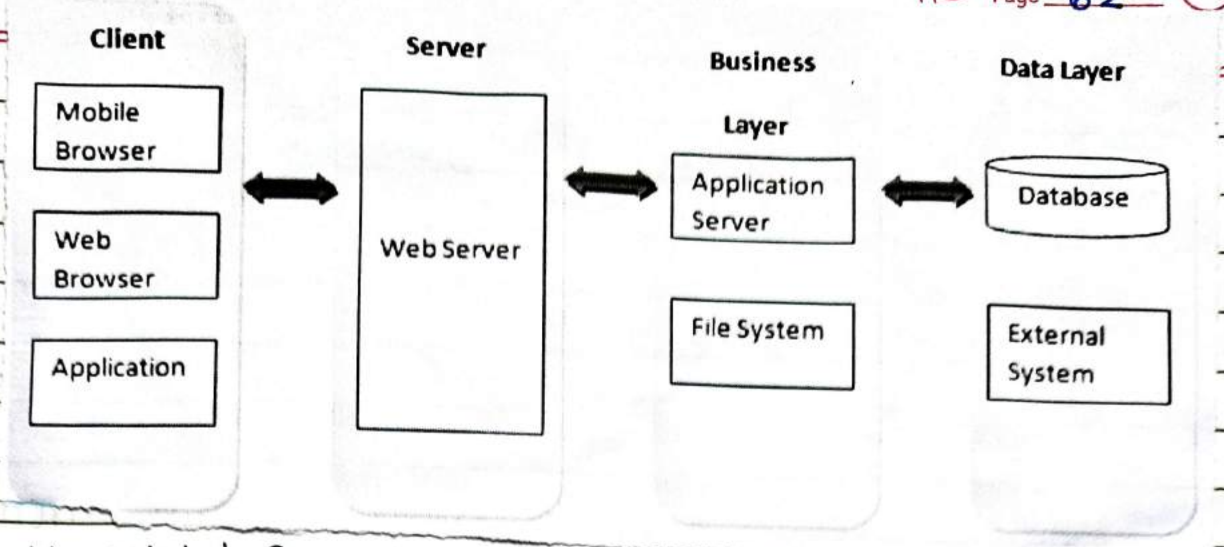
A Web application can be devided in 4 layers:

- Client Layer:
  The client layer contains web browsers, mobile browsers or applications which can make HTTP request to the web server.

- Server Layer:
  The server layer contains web server which can intercepts the request made by clients and pass them the response.

- Business Layer:
  The Business layer contains application server which is utilized by web server to do required processing. This layer interacts with data layer via data base or some external programs.

- Data Layer:
  The data layer contains database or any sour of data.



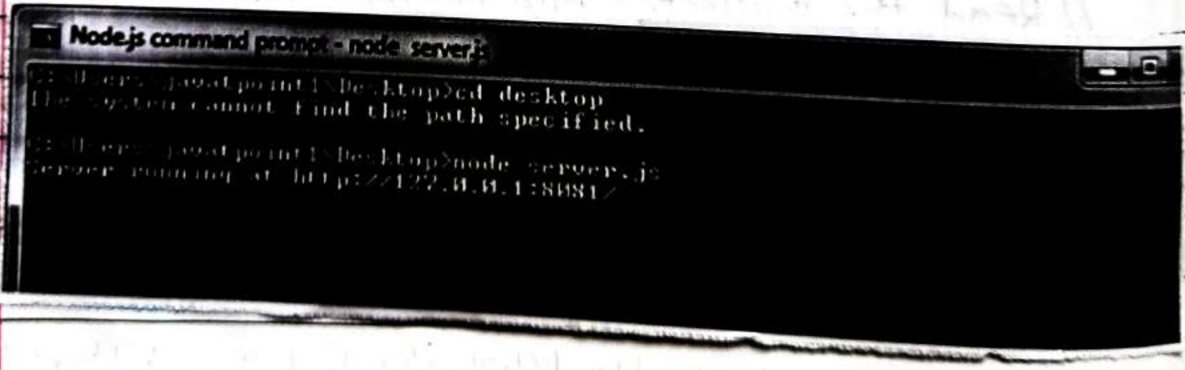
Creating Web Server using Node.js

```
Var http = require('http');
Var fs  = require ('fs');
Varurl  = require('url');
// Create a Server
http. createServer(function(request, response) {
  // Parse the request containing file name
  Var parthname = Url.parse(request.url ).pathname;
  // Print the name of the file for which request is mode.
  Console. log ("Request for" + pathname + "received.");
  // Read the requested file content from file system.
  fs. readFile(pathname. substr(1), function (err. data) {
    if (err) {
      Console. log (err);
      // HTTP Status: 404: NOT FOUND
      // Content Type : text/plain
      response. WriteHead (404, {'Content-Type': 'text/html'});
    } else {
      // Page found
      // HTTP Status : 200: Ok
      // Content Type: text/plain
      response. WriteHead(200, {'Content-Type: 'text/html'});
      // Write the content of the file to response body
       response. Write(data. toString());
    }
```

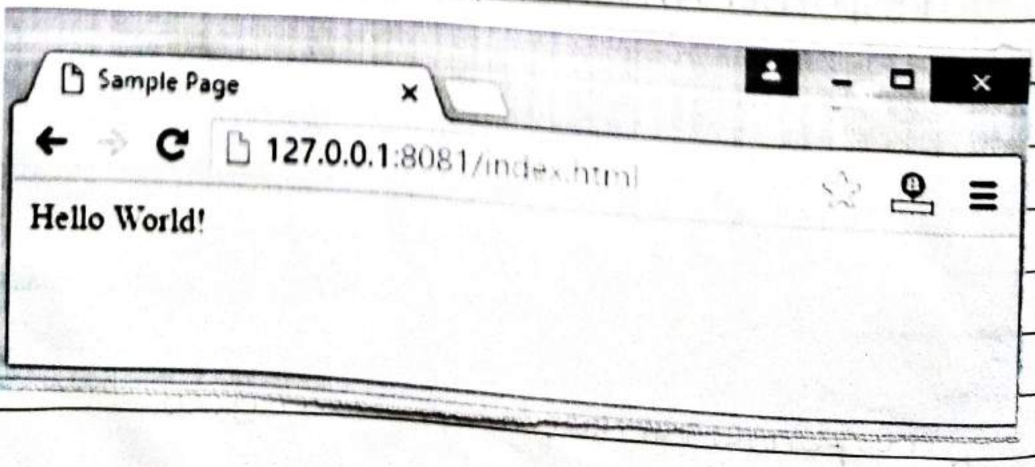
```
        // Send the response body
        response.end();
    });
}).listen(8081);
// console will print the message.
console.log("Server running at http://127.0.0.1:8081/")
```

Next, create an html file named index.html having the following in the same directory where you created server.js

```
<html>
<head>
<title> Sample page </title>
</head>
<body>
Hello World!
</body>
</html>
```



```
Node.js command prompt - node server.js
C:\Users\javatpoint1\Desktop>cd desktop
The system cannot find the path specified.

C:\Users\javatpoint1\Desktop>node server.js
Server running at http://127.0.0.1:8081/
```



Sample Page

127.0.0.1:8081/index.html

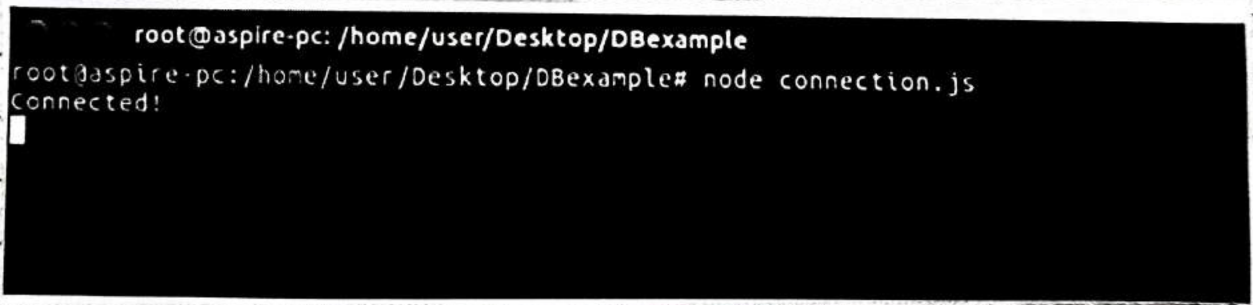
Hello World!

Node.Js Create Connection With MySQL

Create Connection

Create a folder named "DBexample" In that folder Create a js file named "connection.js" having the following code:

```
Var mysql = require('mysql');
 Var con     = mysql.create Connection ({
   host: "localhost",
   User: "root",
   password: "12345"
 });
  con.connect(function(err){
    if(err) throw err;
    Console.log ("Connected!");
 });
```



```
root@aspire-pc: /home/user/Desktop/DBexample
root@aspire-pc:/home/user/Desktop/DBexample# node connection.js
Connected!
```

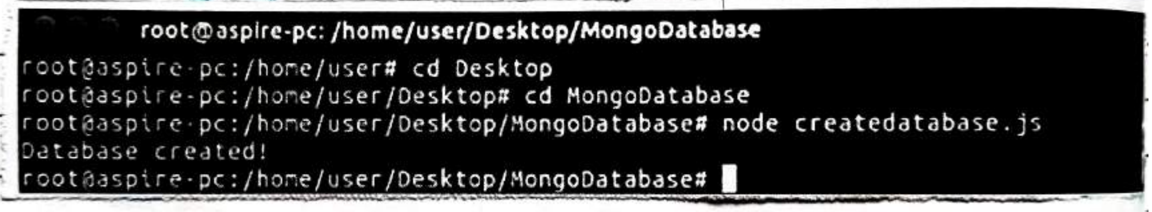
Node.js MySQL Create Database

CREATE DATABASE statement is used to Create a database in MySQL.

Example

For creating a database named "javatpoint".
Create a js file named javatpoint.js having the following data in DBexample folder.

```
Var mysql = require ('my sql');
Var con = mysql.create Connection({
 host: "localhost",
user: "root",
password: "12345"
});
Con.connect(function(err) {
if(err) throw err;
 Console.log("Connected!");
  Con.query("CREATE DATABASE javatpoint", function(err,
                                     result) {
  if (err) throw err;
  Console.log("Connected! ");
  Con.~~query~~ log("Database created ");
  });
  });
```



```
root@aspire-pc: /home/user/Desktop/MongoDatabase
root@aspire-pc:/home/user# cd Desktop
root@aspire-pc:/home/user/Desktop# cd MongoDatabase
root@aspire-pc:/home/user/Desktop/MongoDatabase# node createdatabase.js
Database created!
root@aspire-pc:/home/user/Desktop/MongoDatabase#
```

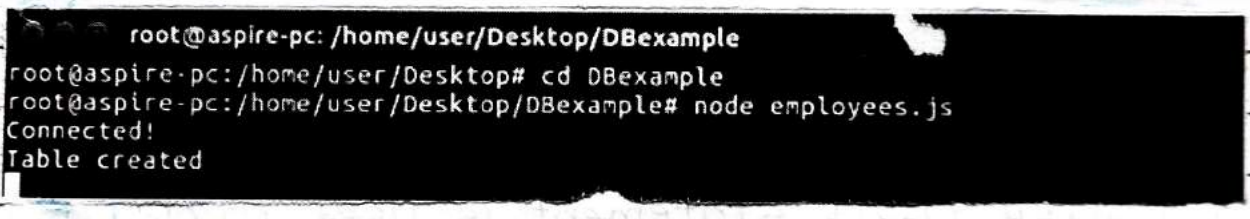
Node.js MySQL Create Table

CREATE TABLE command is used to Create a table in MySQL. You must make it sure that you define the name of the database When you create the Connection.

Example

For creating a table named "employees".
Create a js file named employee.js having the following data in DBexample folder.

```
 Var mysql = require ('mysql');
 Var con = mysql.Create Connection({
 host : "localhost",
user: "root",
password: "12345",
database: "javatpoint"
});
con.connect(function(err){
 if(err) throw err;
 console.log("Connected!");
 Var sql ="CREATE TABLE employees(id INT, name VARCHAR(255),
            age INT(3), city VARCHAR (255))";
 con.query(sql, function (err, result){
  if (err) throw err;
  console.log ("Table created");
  });
 });
```



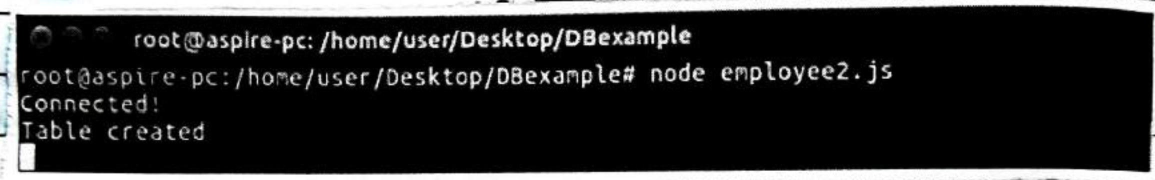
```
root@aspire-pc: /home/user/Desktop/DBexample
root@aspire-pc:/home/user/Desktop# cd DBexample
root@aspire-pc:/home/user/Desktop/DBexample# node employees.js
Connected!
Table created
```

Create Table Having a Primary key

Create Primary key in new table:
Let's create a new table named "employee2" having id as primary key.
Create a js file named employee2.js having the following data in DBexample folder.

```
 Var mysql = require('mysql');
 Var con = mysql.createConnection({
 host: "localhost",
 user: "root",
 password: "12345",
 database: "javatpoint"
});
```

```
Con.connect(function(err) {
if(err) throw err;
 Console.log ("Connected!");
 Var sql = "CREATE TABLE employee 2 (id INT PRIMARY
          KEY, name VARCHAR(255), age INT(3), city VAR
 Con.query (sql, function(err, result) {
  if (err) throw err;
  console.log ("Table created");
  });
  });
```



```
root@aspire-pc: /home/user/Desktop/DBexample
root@aspire-pc:/home/user/Desktop/DBexample# node employee2.js
Connected!
Table created
```

Add columns in existing Table:
ALTER TABLE statement is used to add a column in an existing table. Take the already created table "employee 2" and use a new column salary. Replace the data of the "employee2" table with the following data:

```
 Var mysql = require ("mysql');
  Var Con = mysql.createConnection( {
  host: "localhost";
  user: "root"
   Password: "12345",
   database: "Java t point"
  });
   Con.connect(function (err) {
   if (err) throw err;
   Console.log("Connected!");
```