ASSIGNMENT

Node.Js

Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Styling Guidelines

1. **Whitespace**
   Use 2 space for intending code, One space before the leading braces:
   ```
   Function fn_getId() {
     Console.log('Hai');
   }
   ```
2. **Newlines**
   Newline character (\n) as the last character of a file.

3. **Semicolons**
   Semicolons need to be used before line breaks.

4. **Single quotes**
   Only use single quotes
   ```
   Var strnm='Joe';
   ```

5. **Opening braces on same line**
   Opening braces should go on the same line itself.
   ```
   If (true) {
     Console.log('True');
   }
   ```

6. **Declare one variable per var statement**
   One variable per var statement should be declared to make it easier to re-order the lines.
   ```
   Var  key  =  ['id',  'name'];
   Var  value=  [1,  'Joe'];
   ```

Naming Conventions

1. **Variables, Functions**
   Use lowerCamelCase for variables and function. They should also be descriptive. Single character naming's should be avoided.
   Var firstName = 'Joe Tom';

2. **Class Names**
   Use UpperCamelCase for class names
   class UserProfile() {
   }

3. Don't use trailing or leading underscores

Array Creation

Use trailing commas and put short declaration on single line.

Var array = ['hello', 'world'];

Comments

Use slashes for both single and multi-line comments.

//this is a comment

Functions

Wrap immediately invoked function in parenthesis

```
(function () {
  console.log('welcome');
}());
```

Arrow function

When you use an anonymous function use arrow function notation.

```
[1,2,3].map((x) => {
  const y = x + 1;
return x * y;
});
```

Creating Node.js Application

1. **Import required module:**
   Require directive to load http module and store returned HTTP instance into an http variable.
   var http = require('http');

2. **Creating server:**
   Using created http instance and call http.createServer() method to create a server instance then bind to a port(8081 ,8080) using listen method associated with server instance. Pass function with parameters request and response.

3. **Add HTTP header:**
   If response from http server is to be displayed as HTML should include HTTP header with correct content type.
   res.writeHead**(**200**, {**'Content-Type'**:** 'text/html'**});**

4. **Testing Request & Response:**

   To execute this file node sample.js
   Output will be displayed as **Server running at http://127.0.0.1:8080/**

<u>To display Hello World in web browser:</u>

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

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.end('Hello World!');
}).listen(8080);

console.log('Server running at http://127.0.0.1:8080/');
```

1. File is saved as sample.js
2. Open command prompt and change directory to which node.js program is stored
3. Now run node sample.js on command prompt.
4. Open web browser and check http://localhost:8080

To display current Date:

```
exports.myDateTime = function () {
  return Date();
};
```

This file is stored as myfirstmodule.js

```
var http = require('http');
var dt = require('./myfirstmodule');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write("The date and time are currently: " + dt.myDateTime());
  res.end();
}).listen(8080);
```

This file is stored as fst.js

Now run node fst.js

The output will be The date and time are currently: Thu Apr 29 2021 23:34:05 GMT+0530 (India Standard Time)

Read Query String:

The function http.createServer has a req argument that represent request from client as object, Object has property called url which hold part of url after domain.

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write(req.url);
  res.end();
}).listen(8080);
```

save as demo.js

run node demo.js

http://localhost:8080/ will not return anything

http://localhost:8080/winter will return /winter.

REPL TERMINAL

- **Read** – Reads user's input, parses the input into JavaScript data-structure, and stores in memory.
- **Eval** – Takes and evaluates the data structure.
- **Print** – Prints the result.
- **Loop** – Loops the above command until the user presses **ctrl-c** twice.

Starting REPL

REPL can be started by simply running **node** on shell/console.

C:\WINDOWS\system32\cmd.exe - node

```
Microsoft Windows [Version 10.0.19042.906]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>node
Welcome to Node.js v14.16.0.
Type ".help" for more information.
> 5+2
7
> x=5
5
> y=10
10
> x+y+20+30
65
>
```

Node Package Manager:

Command line utility to install Node.js packages.

npm –version
npm install npm -g
npm install express (to get express use var express = require('express');)

Global Variables:

Global objects are global in nature and available in all modules. These objects are modules, functions, strings and object etc.

List of Node.js global objects are given below:

- __dirname
- __filename
- Console
- Process
- Buffer
- setImmediate(callback[, arg][, ...])
- setInterval(callback, delay[, arg][, ...])
- setTimeout(callback, delay[, arg][, ...])
- clearImmediate(immediateObject)
- clearInterval(intervalObject)
- clearTimeout(timeoutObject)

__dirname

It is a string. It specifies the name of the directory that currently contains the code.

console.log(__dirname);
this is stored in global-example1.js
to run this->node global-example1.js

```
E:\Innovation Incubator\training-repository\Node.js programs>node global-example1.js
E:\Innovation Incubator\training-repository\Node.js programs
```

__filename: It specifies the filename of the code being executed.

Timer:

Set timer functions:

- **setImmediate():** It is used to execute setImmediate.
- **setInterval():** It is used to define a time interval.
- **setTimeout():** ()- It is used to execute a one-time callback after delay milliseconds.

Clear timer functions:

- **clearImmediate(immediateObject):** It is used to stop an immediateObject, as created by setImmediate
- **clearInterval(intervalObject):** It is used to stop an intervalObject, as created by setInterval
- **clearTimeout(timeoutObject):** It prevents a timeoutObject, as created by setTimeout

```
setInterval(function() {
 console.log("setInterval: Hey! 1 millisecond completed!..");
}, 1000);
```

Saved as timer1.js

```
E:\Innovation Incubator\training-repository\Node.js programs>node timer1.js
setInterval: Hey! 1 millisecond completed!..
setInterval: Hey! 1 millisecond completed!..
setInterval: Hey! 1 millisecond completed!..
setInterval: Hey! 1 millisecond completed!..
setInterval: Hey! 1 millisecond completed!..
setInterval: Hey! 1 millisecond completed!..
setInterval: Hey! 1 millisecond completed!..
setInterval: Hey! 1 millisecond completed!..
setInterval: Hey! 1 millisecond completed!..
setInterval: Hey! 1 millisecond completed!..
setInterval: Hey! 1 millisecond completed!..
setInterval: Hey! 1 millisecond completed!..
^C
E:\Innovation Incubator\training-repository\Node.js programs>
```

```
var i =0;
console.log(i);
setInterval(function(){
  i++;
  console.log(i);
}, 1000);
```

This program is saved as timer2.js

```
E:\Innovation Incubator\training-repository\Node.js programs>node timer2.js
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
^C
E:\Innovation Incubator\training-repository\Node.js programs>
```

```
setTimeout(function() {
  console.log("setTimeout: Hey! 1000 millisecond completed!..");
}, 1000);
```

Save as timer3.js

```
E:\Innovation Incubator\training-repository\Node.js programs>node timer3.js
setTimeout: Hey! 1000 millisecond completed!..

E:\Innovation Incubator\training-repository\Node.js programs>
```

Node.js Errors:

The Node.js applications generally face four types of errors:

- **Standard JavaScript errors** i.e. <EvalError>, <SyntaxError>, <RangeError>, <ReferenceError>, <TypeError>, <URIError> etc.
- **System errors**
- **User-specified errors**
- **Assertion errors**

```
try {
  const a = 1;
  const c = a + b;
} catch (err) {
  console.log(err);
}
```

Saved as er1.js

```
E:\Innovation Incubator\training-repository\Node.js programs>node er1.js
ReferenceError: b is not defined
```

Node.js Mysql

Mysql installed using npm install mysql

Create Connection:

```
var mysql = require('mysql');
var con = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: ""
});
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
});
```

```
E:\Innovation Incubator\training-repository\Node.js programs>node ndsql.js
Connected!
^C
```

Create Database:

```
var mysql = require('mysql');
var con = mysql.createConnection({
host: "localhost",
user: "root",
password: "",
database: "mydb1"
});
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;
```

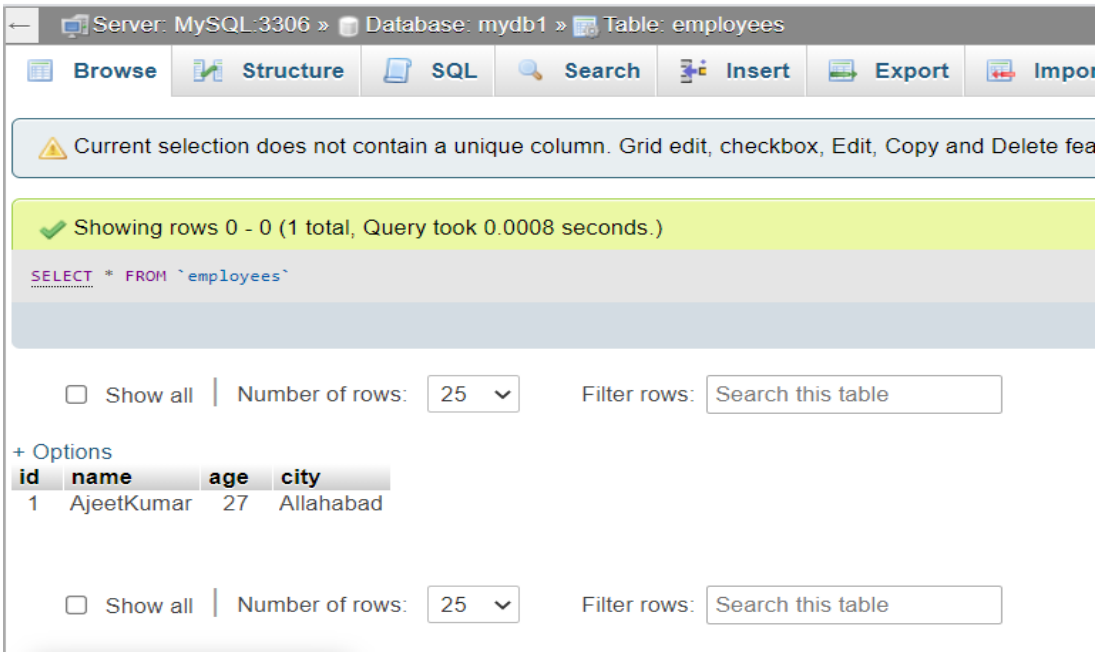
```
console.log("Table created");
});
});
```

```
E:\Innovation Incubator\training-repository\Node.js programs>node ndsql2.js
Connected!
Table created
```

Insert Data:

```
var mysql = require('mysql');
var con = mysql.createConnection({
host: "localhost",
user: "root",
password: "",
database: "mydb1"
});
con.connect(function(err) {
if (err) throw err;
console.log("Connected!");
var sql = "INSERT INTO employees (id, name, age, city) VALUES ('1', 'AjeetKumar', '27', 'Allahabad')";
con.query(sql, function (err, result) {
if (err) throw err;
console.log("1 record inserted");
});
});
```

```
E:\Innovation Incubator\training-repository\Node.js programs>node ndsql3.js
Connected!
1 record inserted
```



Display Data:

```
var mysql = require('mysql');
var con = mysql.createConnection({
host: "localhost",
user: "root",
password: "",
database: "mydb1"
});
con.connect(function(err) {
if (err) throw err;
con.query("SELECT * FROM employees", function (err, result) {
if (err) throw err;
console.log(result);
});
});
```

```
E:\Innovation Incubator\training-repository\Node.js programs>node ndsql4.js
[
  RowDataPacket {
    id: 1,
    name: 'AjeetKumar',
    age: 27,
    city: 'Allahabad'
  }
]
```

```
var mysql = require('mysql');
var con = mysql.createConnection({
host: "localhost",
user: "root",
password: "",
database: "mydbnad"
});
con.connect(function(err) {
if (err) throw err;
con.query("SELECT * FROM tbl_log", function (err, result) {
if (err) throw err;
console.log(result);
});
});
```

```
E:\Innovation Incubator\training-repository\Node.js programs>node ndsql.js
[
  RowDataPacket {
    logid: 1,
    usname: 'ADMIN',
    passwd: '3f7caa3d471688b704b73e9a77b1107f',
    role: 'admin',
    logsts: 1
  },
  RowDataPacket {
    logid: 2,
    usname: 'Joet12',
    passwd: 'dcdaa8a48027342dce14001471658a40',
    role: 'customer',
    logsts: 1
  },
  RowDataPacket {
    logid: 3,
    usname: 'emp1',
    passwd: '0729a29331ba83a71322e69d2f2ed514',
    role: 'employee',
    logsts: 1
  },
  RowDataPacket {
    logid: 4,
    usname: 'emp2',
    passwd: 'd04faaafa7d63255056677b5839ec91f',
    role: 'employee',
    logsts: 1
  }
]
^C
```

Jest Framework for unit testing

To install jest use **npm install --save-dev jest**

```
function sum(a, b) {
  return a + b;
}
module.exports = sum;
```

save this file as **sum.js**

now write the actual test

```
const sum = require('./sum');

test('adds 1 + 2 to equal 3', () => {
  expect(sum(1, 2)).toBe(3);
});
```

Save it as **sum.jest.js**

Add following in **package.json**

```
{
  "scripts": {
    "test": "jest"
  }
}
```

Now to run test npm run test

```
E:\Innovation Incubator\training-repository\Node.js programs>npm test

> @ test E:\Innovation Incubator\training-repository\Node.js programs
> jest

 PASS  ./sum.test.js
  √ adds 1 + 2 to equal 3 (3 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        15.849 s
Ran all test suites.

E:\Innovation Incubator\training-repository\Node.js programs>
```

When error occurs changed – to * in sum.js

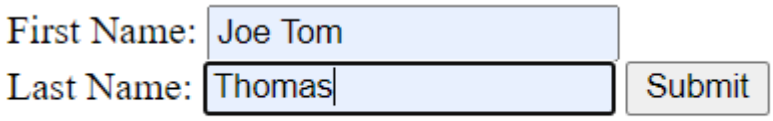
```
E:\Innovation Incubator\training-repository\Node.js programs>npm test

> @ test E:\Innovation Incubator\training-repository\Node.js programs
> jest

 FAIL  ./sum.test.js
  × adds 2 + 3 to equal 5 (6 ms)

  ● adds 2 + 3 to equal 5

    expect(received).toBe(expected) // Object.is equality

    Expected: 5
    Received: 6

      2 |
      3 | test('adds 2 + 3 to equal 5', () => {
    > 4 |   expect(sum(2, 3)).toBe(5);
        |                     ^
      5 | });

      at Object.<anonymous> (sum.test.js:4:21)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 1 total
Snapshots:   0 total
Time:        1.836 s
Ran all test suites.
npm ERR! Test failed.  See above for more details.

E:\Innovation Incubator\training-repository\Node.js programs>
```

Express.js

Express.js is a web framework for Node.js. It is a fast, robust and asynchronous in nature.

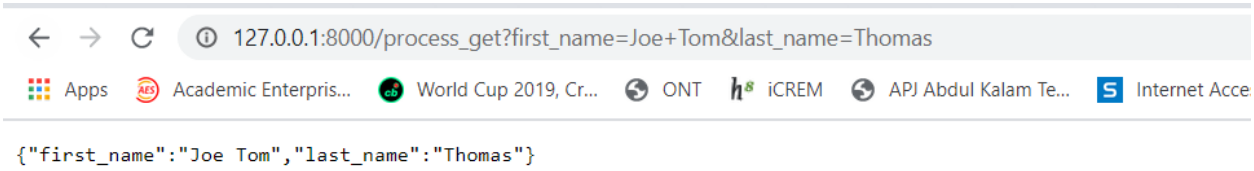
Now create html file index.html

```
<html>
<body>
<form action="http://127.0.0.1:8000/process_get" method="GET">
First Name: <input type="text" name="first_name">  <br>
Last Name: <input type="text" name="last_name">
<input type="submit" value="Submit">
</form>
</body>
</html>
```




```
var express = require('express');
var app = express();
app.use(express.static('public'));

app.get('/index.html', function (req, res) {
  res.sendFile( __dirname + "/" + "index.html" );
})
app.get('/process_get', function (req, res) {
response = {
      first_name:req.query.first_name,
      last_name:req.query.last_name
  };
  console.log(response);
  res.end(JSON.stringify(response));
```

```
})
var server = app.listen(8000, function () {

  var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at http://%s:%s", host, port)

})
```



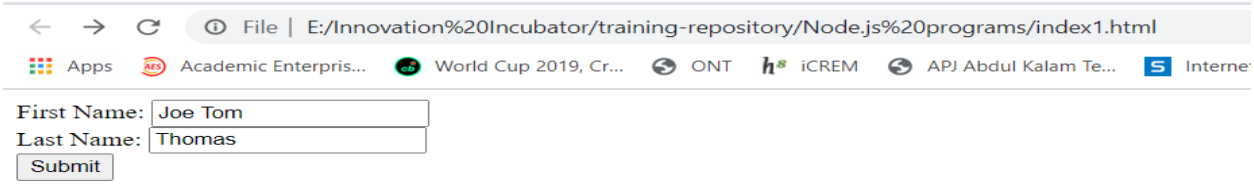

```
{"first_name":"Joe Tom","last_name":"Thomas"}
```

```
E:\Innovation Incubator\training-repository\Node.js programs>node get_example1.js
Example app listening at http://:::8000
{ first_name: 'Joe Tom', last_name: 'Thomas' }
```

Fetch data in paragraph format

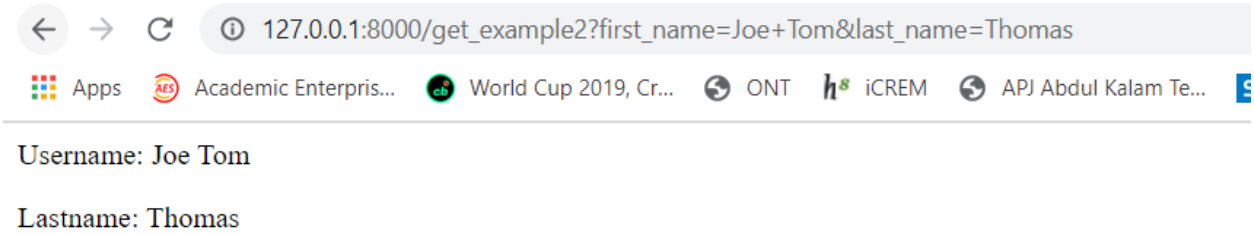
Save as index1.html

```
<html>
<body>
<form action="http://127.0.0.1:8000/get_example2" method="GET">
First Name: <input type="text" name="first_name"/>  <br/>
Last Name: <input type="text" name="last_name"/><br/>
<input type="submit" value="Submit"/>
</form>
</body>
</html>
```



Save as **get_example2.js**

```
var express = require('express');
var app=express();
app.get('/get_example2', function (req, res) {
res.send('<p>Username: ' + req.query['first_name']+'</p><p>Lastname: '+req.query['last_name']+'</p>');
})
var server = app.listen(8000, function () {
  var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at http://%s:%s", host, port)
})
```



Mysql on Express:

```
<html>
<body>
<form action="/submit" method="post">
First Name: <input type="text" name="first_name"/>  <br/>
Last Name: <input type="text" name="last_name"/><br/>
<input type="submit" id="submit" name="submit"/>
</form>
</body>
</html>
```

Save as index.html

Saved file as hello.js

```
const express = require('express')
const bodyParser =require('body-parser')
var mysql = require('mysql');
const app = express()
const port = 3000


app.use(bodyParser.urlencoded({extended: false}))
app.set('view engine', 'pug')
app.get('/', function (req, res) {
  res.sendFile('index.html', { root:__dirname })
});

var con = mysql.createConnection({
host: "localhost",
user: "root",
```

```
password: "",
database: "mydb1"
});

con.connect(function(err) {
if (err) throw err;
  console.log("Connected!");
});
app.post('/submit',  function  (req,  res)  {
    console.log(req.body);
    var sql="insert into nme values('"+req.body.first_name+"',  '"+req.body.last_name+"')";
    con.query(sql,  function  (err,   rows,  fields){
    if (err) throw err
      console.log('1 user added');
    });
    con.end();
});
app.listen(port,  ()  =>  console.log('Example app listening on port ${port}!'))
```

to run this file type node hello.js

```
E:\Innovation Incubator\training-repository\Node.js programs>node hello.js
Example app listening on port ${port}!
Connected!
[Object: null prototype] {
  first_name: 'Joe ',
  last_name: ' Tom Thomas',
  submit: 'Submit'
}
1 user added
^C
E:\Innovation Incubator\training-repository\Node.js programs>
```

Server: MySQL:3306 » Database: mydb1 » Table: nme

Browse | Structure | SQL | Search | Insert | Export | Import

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete feat

Showing rows 0 - 1 (2 total, Query took 0.0252 seconds.)

```
SELECT * FROM `nme`
```

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

fnme	lnme
Joe Tom	Thomas
Joe	Tom Thomas