

MAYURI SHRIDATTA YERANDE

D15B-ROLL NO: 70

INTERNET PROGRAMMING

EXPERIMENT 10

Aim:- Experiment to study basics Node Js.

Theory:

Node.js is a cross-platform runtime environment and library for running JavaScript applications outside the browser. It is used for creating server-side and networking web applications. It is open source and free to use.

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js also provides a rich library of various JavaScript modules to simplify the development of web applications.

Features of Node.js

Following is a list of some important features of Node.js that makes it the first choice of software architects.

1. Extremely fast: Node.js is built on Google Chrome's V8 JavaScript Engine, so its library is very fast in code execution.
2. I/O is Asynchronous and Event Driven: All APIs of Node.js library are asynchronous i.e. non-blocking. So a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call. It is also a reason that it is very fast.
3. Single threaded: Node.js follows a single threaded model with event looping.
4. Highly Scalable: Node.js is highly scalable because the event mechanism helps the server to respond in a non-blocking way.
5. No buffering: Node.js cuts down the overall processing time while uploading audio and video files. Node.js applications never buffer any data. These applications simply output the data in chunks.
6. Open source: Node.js has an open source community which has produced many excellent modules to add additional capabilities to Node.js applications.

7. License: Node.js is released under the MIT license.

Node.js is a cross-platform JavaScript runtime environment. It allows the creation of scalable Web servers without threading and networking tools using JavaScript and a collection of “modules” that handle various core functionalities. It can make console-based and web-based node.js applications.

Loose Typing: Node.js supports loose typing, it means you don’t need to specify what type of information will be stored in a variable in advance. We use the var keyword in Node.js to declare any type of variable. Examples are given below:

Objects: Node.js objects are the same as JavaScript objects i.e. the objects are similar to variables and it contains many values which are written as name : value pairs. Name and value are separated by colon and every pair is separated by comma.

Third-party modules: Third-party modules are modules that are available online using the Node Package Manager(NPM). These modules can be installed in the project folder or globally. Some of the popular third-party modules are mongoose, express, angular, and react.

```
npm install express
```

```
npm install mongoose
```

```
npm install -g @angular/cli
```

Functions: Node.js functions are defined using the function keyword then the name of the function and parameters which are passed in the function. In Node.js, we don’t have to specify data types for the parameters and check the number of arguments received. Node.js functions follow every rule which is there while writing JavaScript functions.

String and String Functions: In Node.js we can make a variable as string by assigning a value either by using single (”) or double (“”) quotes and it contains many functions to manipulate to strings.

Node.js REPL (READ, EVAL, PRINT, LOOP):

REPL (READ, EVAL, PRINT, LOOP) is a computer environment similar to Shell (Unix/Linux) and command prompt. Node comes with the REPL environment when it is installed. System interacts with the user through outputs of commands/expressions used. It is useful in writing and debugging the codes. The work of REPL can be understood from its full form:

- Read : It reads the inputs from users and parses it into JavaScript data structure. It is then stored in memory.
- Eval : The parsed JavaScript data structure is evaluated for the results.
- Print : The result is printed after the evaluation.
- Loop : Loops the input command.

Getting Started with REPL: To start working with the REPL environment of NODE; open up the terminal (in case of UNIX/LINUX) or the Command prompt (in case of Windows) and write node and press 'enter' to start the REPL.

```
PS D:\Sem 5 assignment\IP\mayuri_EXP-10> node
Welcome to Node.js v16.18.0.
Type ".help" for more information.
> a=2
2
> b=3
3
> a+b
5
> a-b
-1
> 
```

Node.js web-based: A node.js web application contains the following three parts:

1. Import required modules: The "require" directive is used to load a Node.js module.
2. Create server: You have to establish a server which will listen to client's request similar to Apache HTTP Server.
3. Read request and return response: Server created in the second step will read HTTP request made by client which can be a browser or console and return the response.

Node.js Modules: In Node.js, Modules are the blocks of encapsulated code that communicate with an external application on the basis of their related functionality. Modules can be a single file or a collection of multiple files/folders. The reason programmers are heavily reliant on modules is because of their re-usability as well as the ability to break down a complex piece of code into manageable chunks.

Modules are of three types:

- Core Modules
- local Modules
- Third-party Modules

Node.js File System (FS):

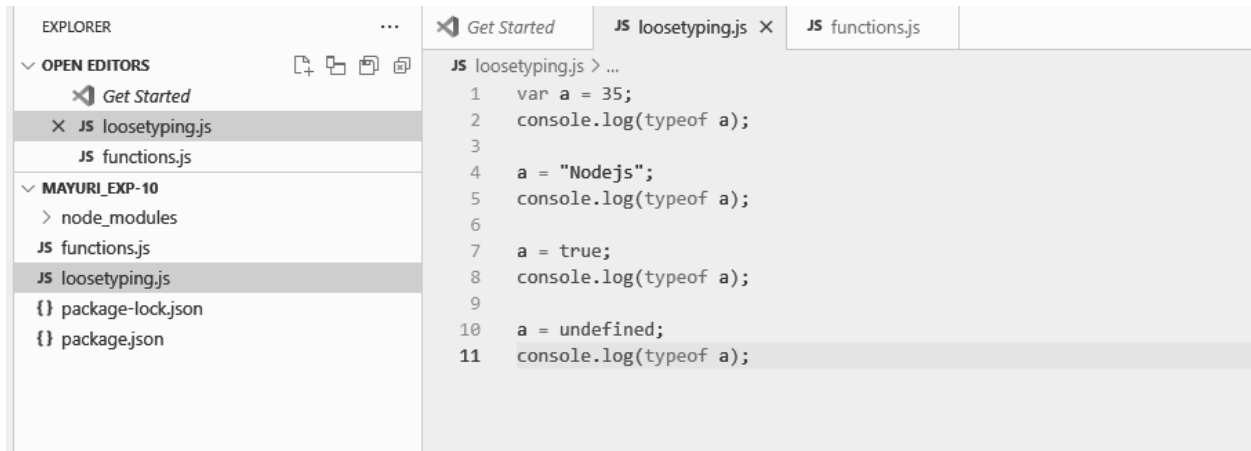
In Node.js, file I/O is provided by simple wrappers around standard POSIX functions. Node File System (fs) module can be imported using following syntax:

Asynchronous methods take a last parameter as completion function callback. Asynchronous method is preferred over synchronous method because it never blocks the program execution where as the synchronous method blocks.

Implementation:

LOOSE TYPING EXAMPLE

CODE



```
EXPLORER
  OPEN EDITORS
    Get Started
    JS loosetyping.js
    JS functions.js
  MAYURI_EXP-10
    > node_modules
    JS functions.js
    JS loosetyping.js
    {} package-lock.json
    {} package.json

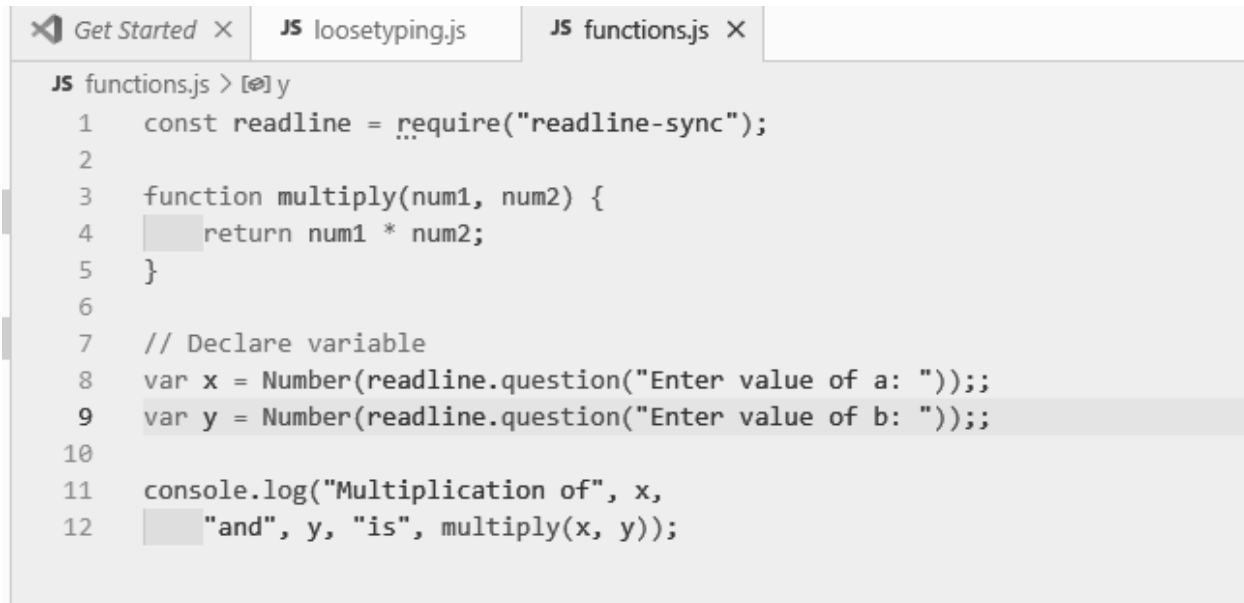
JS loosetyping.js > ...
1  var a = 35;
2  console.log(typeof a);
3
4  a = "Nodejs";
5  console.log(typeof a);
6
7  a = true;
8  console.log(typeof a);
9
10 a = undefined;
11 console.log(typeof a);
```

Output:

```
PS D:\Sem 5 assignment\IP\mayuri_EXP-10> node loosetyping.js
number
string
boolean
undefined
PS D:\Sem 5 assignment\IP\mayuri_EXP-10> █
```

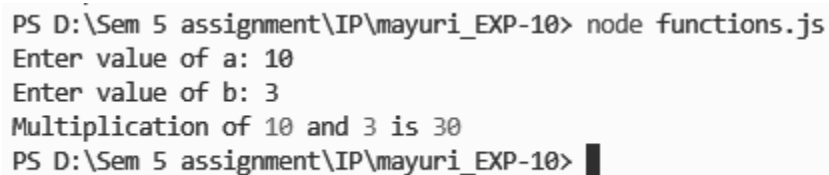
FUNCTIONS EXAMPLE:

CODE:

A screenshot of a code editor with three tabs: 'Get Started', 'JS looosetyping.js', and 'JS functions.js'. The 'JS functions.js' tab is active, showing the following code:

```
JS functions.js > [0] y
1  const readline = require("readline-sync");
2
3  function multiply(num1, num2) {
4    return num1 * num2;
5  }
6
7  // Declare variable
8  var x = Number(readline.question("Enter value of a: "));
9  var y = Number(readline.question("Enter value of b: "));
10
11 console.log("Multiplication of", x,
12            "and", y, "is", multiply(x, y));
```

Output:

A screenshot of a terminal window showing the execution of the script. The prompt is 'PS D:\Sem 5 assignment\IP\mayuri_EXP-10>'. The user enters 'node functions.js'. The output is:

```
PS D:\Sem 5 assignment\IP\mayuri_EXP-10> node functions.js
Enter value of a: 10
Enter value of b: 3
Multiplication of 10 and 3 is 30
PS D:\Sem 5 assignment\IP\mayuri_EXP-10> █
```

CALCULATOR EXAMPLE

Filename: calci.js

```
const readline = require("readline-sync");
```

```
var x = Number(readline.question("Enter value of a: "));
var y = Number(readline.question("Enter value of b: "));
```

```
add = function(x, y) {
  return x + y;
};
sub = function(x, y) {
  return x - y;
};
```

```
mult = function(x, y) {
```

```

    return x * y;
};

div = function(x, y) {
    return x / y;
};

console.log("Addition: " + add(x, y));

console.log("Subtraction: " + sub(x, y));

console.log("Multiplication: " + mult(x, y));

console.log("Division: " + div(x, y));

```

OUTPUT:

```

PS D:\Sem 5 assignment\IP\mayuri_EXP-10> node calci.js
Enter value of a: 10
Enter value of b: 5
Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2
PS D:\Sem 5 assignment\IP\mayuri_EXP-10> 

```

WEB BASED EXAMPLE

CODE:

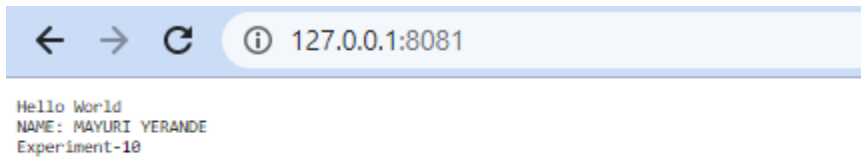
```

JS helloworld.js > ...
1  const http = require('http');
2  http.createServer(function(request, response) {
3      response.writeHead(200, { 'Content-Type': 'text/plain' });
4      response.end('Hello World\nNAME: MAYURI YERANDE\nExperiment-10');
5  }).listen(8081);
6  console.log('Server running at http://127.0.0.1:8081/')

```

OUTPUT:

```
PS D:\Sem 5 assignment\IP\mayuri_EXP-10> node helloworld.js
Server running at http://127.0.0.1:8081/
```



EXAMPLE-2:

CODE:

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

var server = http.createServer(function(req, res) {
  if (req.url === '/') {

    res.writeHead(200, { 'Content-Type': 'text/html' });

    res.write("<html><body><p>Welcome to Mayuri's home Page.</p></body></html>");
    res.end();

  } else if (req.url === "/student") {

    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.write('<html><body><p>This is student Page.\n Name: MAYURI YERANDE\n ROLL NO:-70</p></body></html>');
    res.end();

  } else if (req.url === "/admin") {

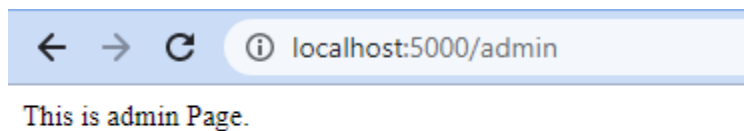
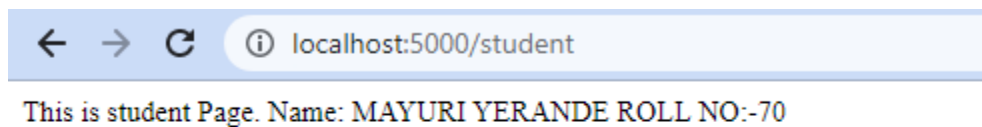
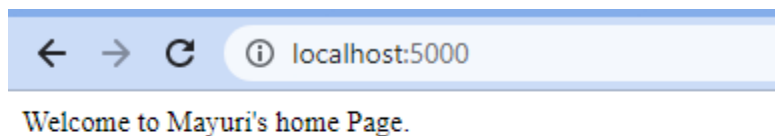
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.write('<html><body><p>This is admin Page.</p></body></html>');
    res.end();

  } else
```

```
res.end('Invalid Request!');  
});  
server.listen(5000);  
console.log('Node.js web server at http://localhost:5000/ is running..')
```

OUTPUT:

```
PS D:\Sem 5 assignment\IP\mayuri_EXP-10> node server.js  
Node.js web server at http://localhost:5000/ is running..  
█
```



NODE JS FILE SYSTEM EXAMPLE

CODE:

```
input.txt
1  hello
```

```
JS read.js > ...
1  var fs = require("fs");
2
3  // Asynchronous read
4  fs.readFile('input.txt', function(err, data) {
5      if (err) {
6          return console.error(err);
7      }
8      console.log("Asynchronous read: " + data.toString());
9  });
10
11 // Synchronous read
12 var data = fs.readFileSync('input.txt');
13 console.log("Synchronous read: " + data.toString());
14
15 console.log("Program Ended");
```

OUTPUT:

```
PS D:\Sem 5 assignment\IP\mayuri_EXP-10> node read.js
Synchronous read: hello
Program Ended
Asynchronous read: hello
PS D:\Sem 5 assignment\IP\mayuri_EXP-10> █
```

Conclusion: We have successfully implemented basic codes in Nodejs.