D15B/37 **Mikil Lalwani** **IP Lab 9**

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 the 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 that 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.
Datatypes: Node.js contains various types of data types similar to JavaScript.

- Boolean
- Undefined

- Null
- String
- Number

Loose Typing: Node.js supports loose typing, which 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:
Example:

```
// Variable store number data type
var a = 35;
console.log(typeof a);

// Variable store string data type
a = "Nodejs";
console.log(typeof a);

// Variable store Boolean data type
a = true;
console.log(typeof a);

// Variable store undefined (no value) data type
a = undefined;
console.log(typeof a);
```

Output:
number
string
boolean
undefined

Objects & Functions:

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 a colon and every pair is separated by a comma.
Example:

```
  var company = {
     Name: "Nodejs",
     Address: "Mumbai",
     Contact: "+123456670",
     Email: "abc@xyz.org"
  };

  // Display the object information
  console.log("Information of variable company:", company);

  // Display the type of variable
  console.log("Type of variable company:", typeof company);
```

Output:

```
Information of variable company: {
  Name: "Nodejs",
    Address: "Mumbai",
    Contact: "+123456670",
    Email: "abc@xyz.org"
};
```

Type of variable company: object

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.

Example:

```
function multiply(num1, num2) {

    // It returns the multiplication
    // of num1 and num2
    return num1 * num2;
}

// Declare variable
var x = 2;
var y = 3;

// Display the answer returned by
// multiply function
console.log("Multiplication of", x,
    "and", y, "is", multiply(x, y));
```

Output:

Multiplication of 2 and 3 is 6

If you observe in the above example, we have created a function called “multiply” with parameters the same like JavaScript.
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.
Following is the example of defining string variables and functions in node.js.
Example:

```
var x = "Welcome to learning Nodejs ";

var y = 'Node.js Tutorials';

var z = ['Nodejs', 'for', 'Node'];

console.log(x);

console.log(y);

console.log("Concat Using (+) :", (x + y));

console.log("Concat Using Function :", (x.concat(y)));

console.log("Split string: ", x.split(' '));

console.log("Join string: ", z.join(', '));

console.log("Char At Index 5: ", x.charAt(5) );
```

Output:
Welcome to Nodejs
Node.js Tutorials
Concat Using (+) : Welcome to Nodejs Node.js Tutorials
Concat Using Function : Welcome to Nodejs Node.js Tutorials
Split string: ['Welcome', 'to', 'Nodejs', '']
Join string: Nodejs, for, Node
Char At Index 5: j

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. The system interacts with the user through outputs of commands/expressions used. It is useful in writing and debugging codes. The work of REPL can be understood from its full form:
Read: It reads the inputs from users and parses them 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.

```
> i=0;
0
> do{
... cosole.log(i);
... i++;
... }while(i<=5);
Uncaught ReferenceError: cosole is not defined
> do{
... console.log(i);
... i++;
... }while(i<=5);
0
1
2
3
4
5
5
>
```

There can be console-based and web-based node.js applications.

Node.js console-based Example

File: console_example1.js

```
 console. log('Hello JavaTpoint');
```

Open Node.js command prompt and run the following code:

```
$ node console_example1.js
```

Conclusion-

Thus we have successfully implemented the basics of Node JS.