

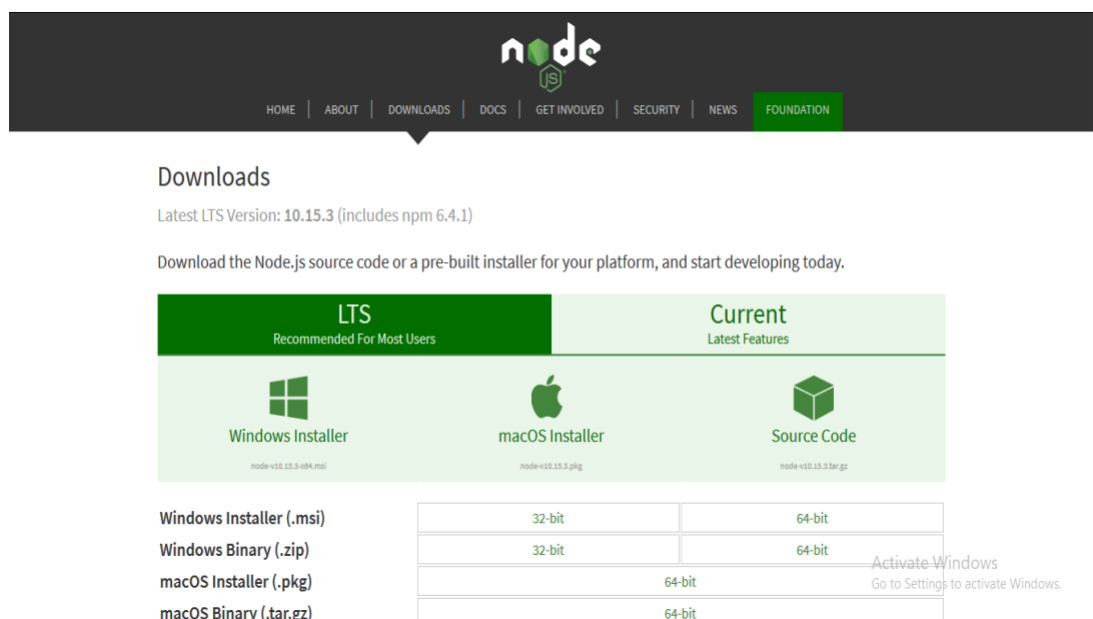
PROGRAM 3:

Aim: Write node.js program to create, access, modify Arrays.

Installation of Node.js on Windows

Step-1: Downloading the Node.js '.msi' installer.

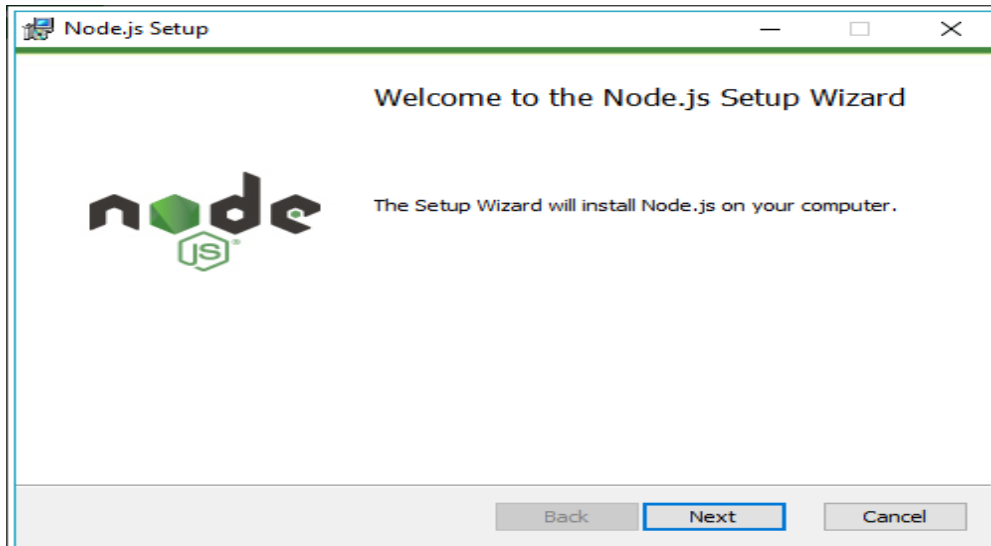
The first step to install Node.js on windows is to download the installer. Visit the official Node.js website i.e) <https://nodejs.org/en/download/> and download the .msi file according to your system environment (32-bit & 64-bit). An MSI installer will be downloaded on your system.



Step-2: Running the Node.js installer.

Now you need to install the node.js installer on your PC. You need to follow the following steps for the Node.js to be installed:-

- Double click on the .msi installer.
The Node.js Setup wizard will open.
- Welcome To Node.js Setup Wizard.
Select "Next"



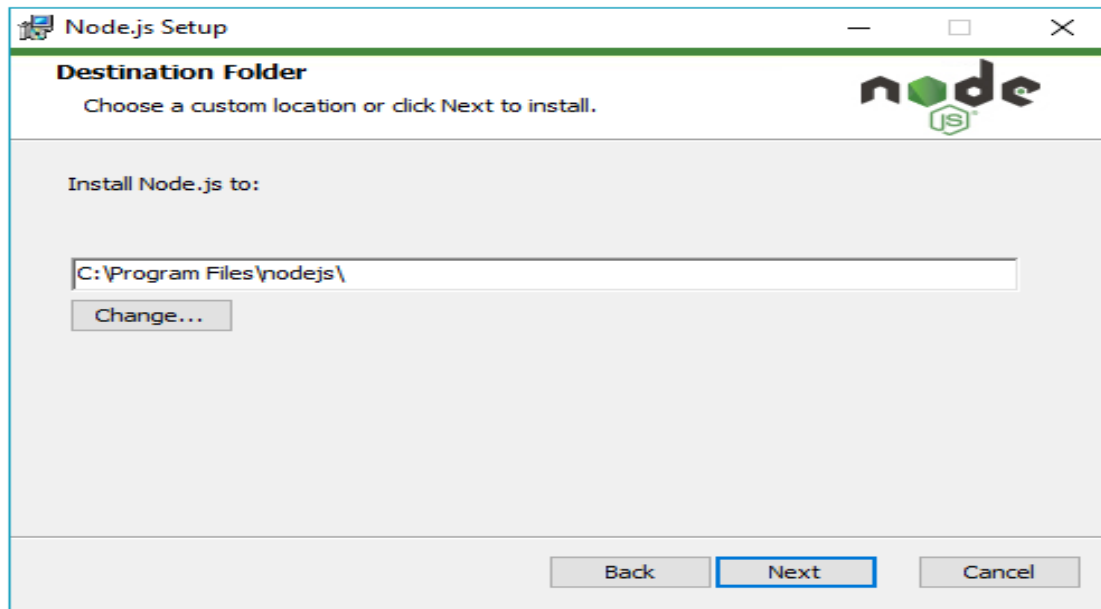
After clicking “Next”, End-User License Agreement (EULA) will open

**Check “I accept the terms in the License Agreement”
Select “Next”**

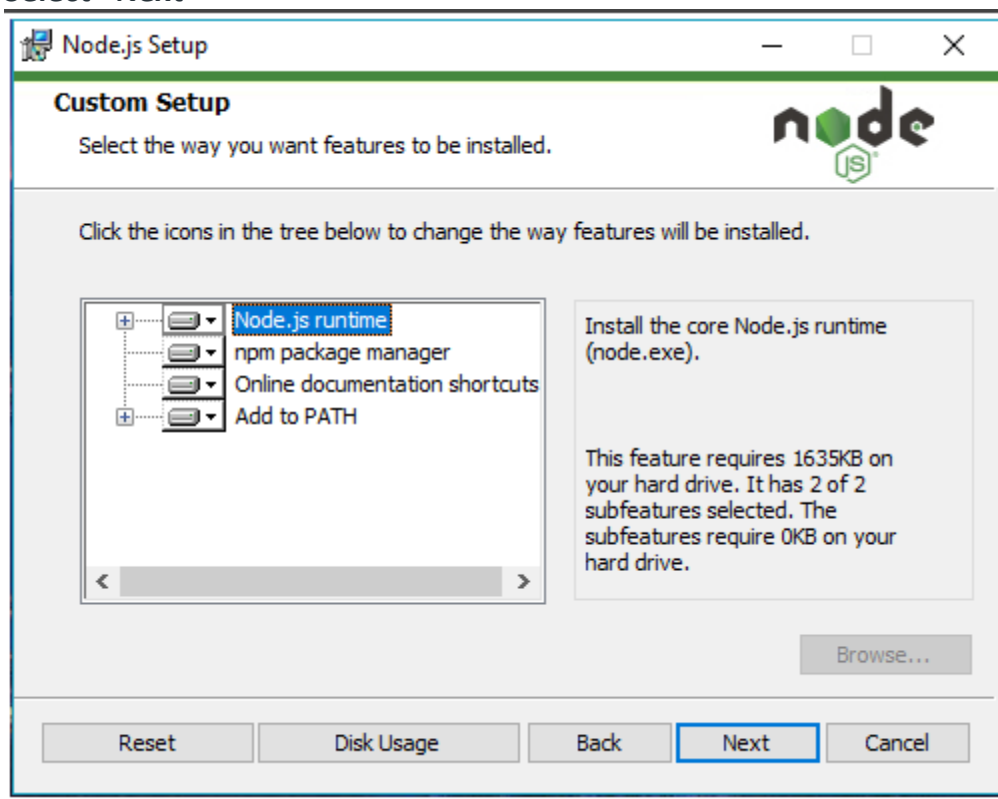


Destination Folder

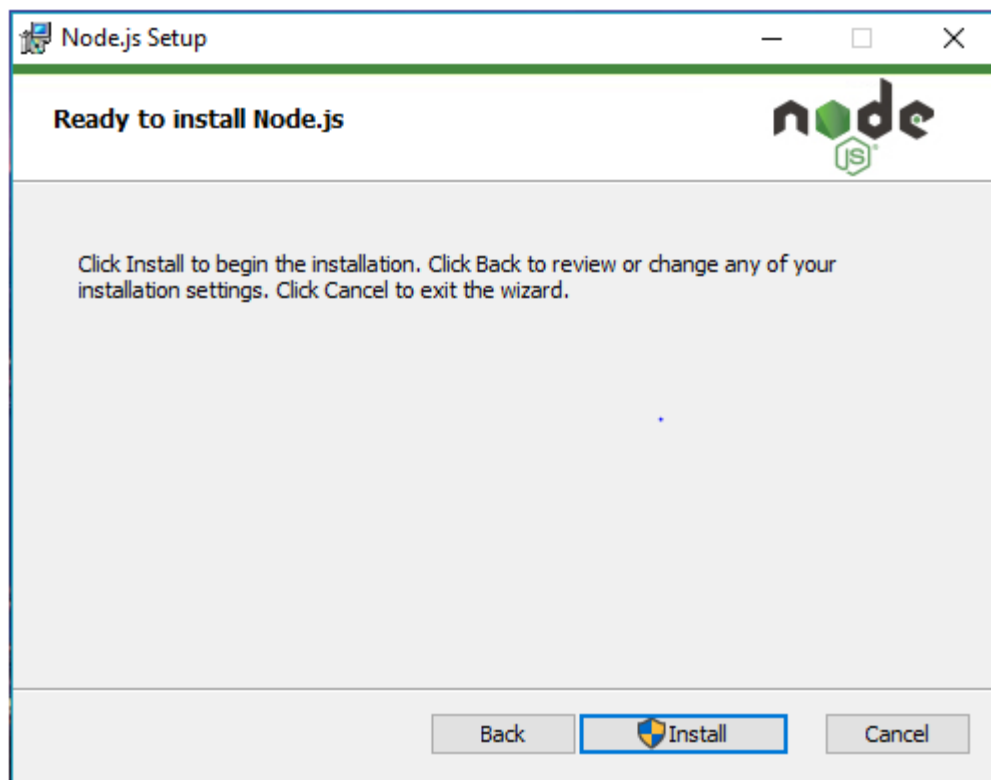
Set the Destination Folder where you want to install Node.js & Select “Next”



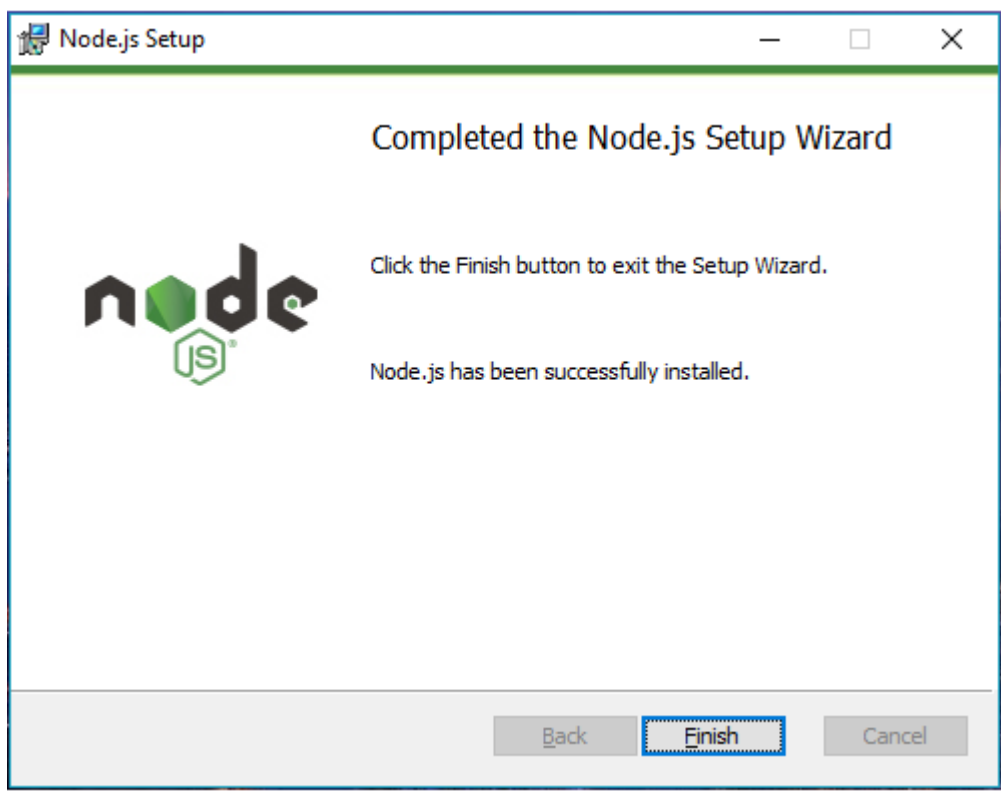
Custom Setup
Select "Next"



Ready to Install Node.js.
Select "Install"



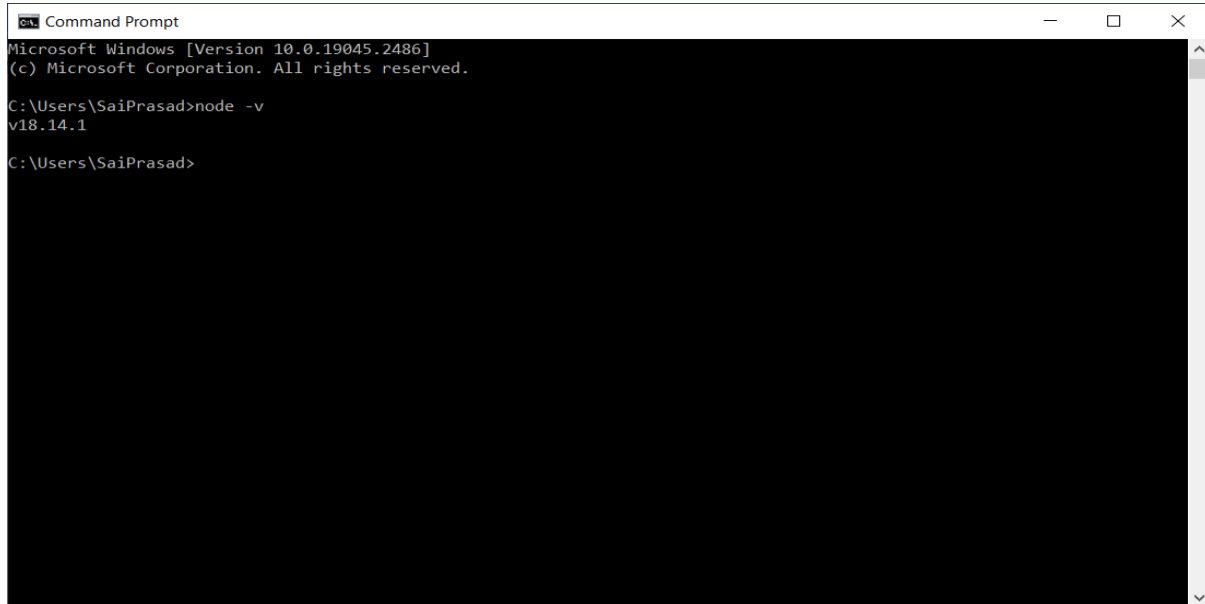
Click "Finish"



Step 3: Verify that Node.js was properly installed or not.

To check that node.js was completely installed on your system or not, you can run the following command in your command prompt or Windows Powershell and test it:-

C:\Users\Admin> node -v

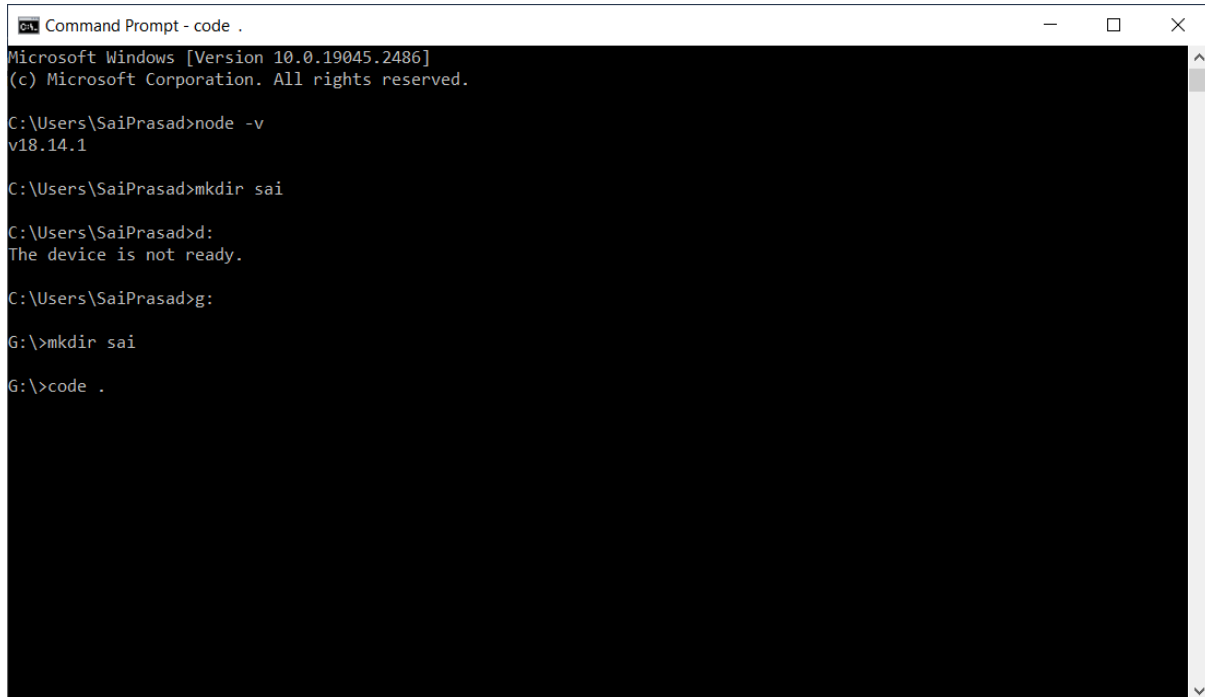


```
Command Prompt
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SaiPrasad>node -v
v18.14.1

C:\Users\SaiPrasad>
```

Create Directory



```
Command Prompt - code .
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SaiPrasad>node -v
v18.14.1

C:\Users\SaiPrasad>mkdir sai

C:\Users\SaiPrasad>d:
The device is not ready.

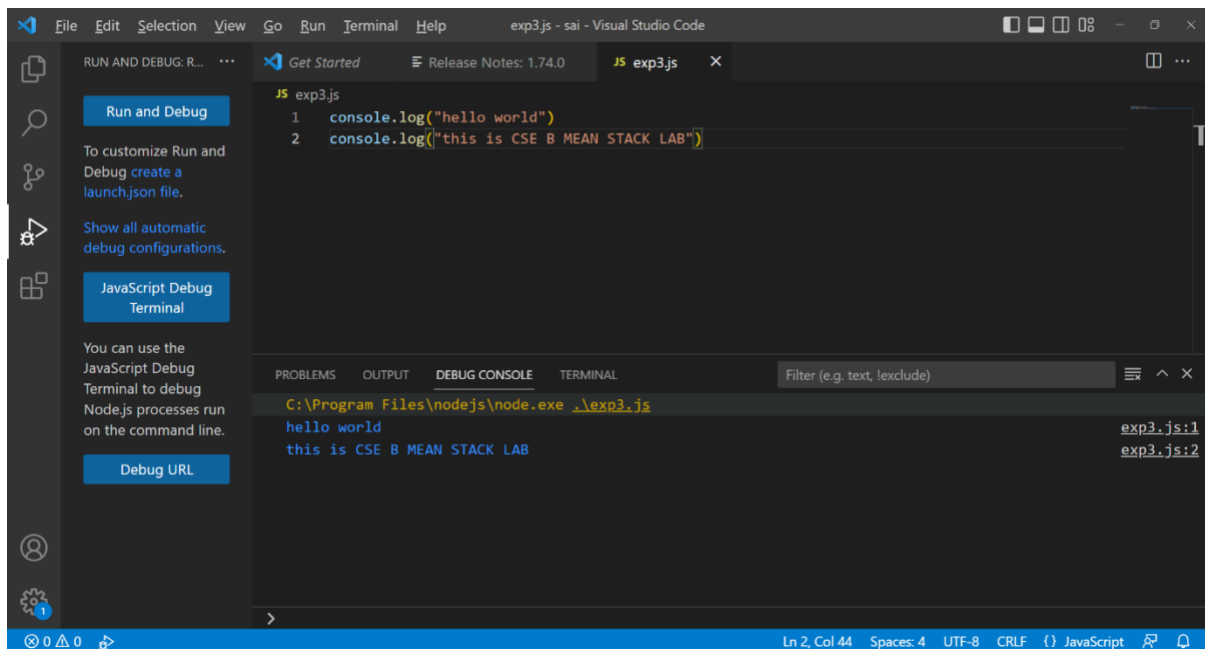
C:\Users\SaiPrasad>g:

G:\>mkdir sai

G:\>code .
```

Visual studio environment will be opened with the command code .

“Enter”



Array:

An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Create Arrays in Nodejs:

We can create the arrays in nodejs with the following code. To create an empty array

```
var array = [];
```

To create an array with elements

```
var arr = [ 'cat', 'goat', 'bat' ];
```

Array Functions in Node js:

Index:

Access the array, we can access the nodejs array using the index of the array, array index starts from 0. Index number is used along with the [] operator.

```
var arr = [ 'cat', 'goat', 'bat' ];  
console.log(arr[0]) // prints cat  
console.log(arr[2]) // prints bat
```

INDEXOF:

indexOf method is used to return the first index of the element passed within the array or otherwise it will return -1 if the value is not found.

```
var arr = [ 'cat', 'dog', 'goat', 'mummy', 'goat' ];
console.log(arr) // prints 'cat', 'dog', 'goat', 'mummy', 'goat'
var position = arr.indexOf("avatar");
console.log(position)
//prints -1
console.log(arr.indexOf("goat"))
//prints 2
```

lastIndexOf() :

lastIndexOf method is used to return the last index of the element passed within the array or otherwise it will return -1 if the value is not found.

```
var arr = [ 'cat', 'dog', 'goat', 'cat'];
console.log(arr) // prints 'cat', 'dog', 'goat', 'cat'
var position = arr.lastIndexOf("avatar");
console.log(position)
//prints -1
console.log(arr.lastIndexOf("goat"))
//prints 3
```

push function :

push() function helps to create an object inside the array,

```
const fruits = [];
fruits.push("banana", "apple", "peach");
console.log(fruits.length); // prints '3'
console.log(fruits); //prints "banana", "apple", "peach";
fruits.push("grapes");
console.log(fruits); //prints 'banana', 'apple', 'peach', 'grapes'
console.log(fruits.pop()); // prints 'grapes'
```

pop function :

pop() function removes the element from the end of the array. pop() does not accept any parameter

```
var arr = [ 'cat', 'dog', 'goat' ];
console.log(arr) // prints 'cat', 'dog', 'goat'
```

```
arr.pop()
console.log(arr) // prints 'cat', 'dog'
```

unshift function :

unshift() method in arrays is used to insert an element at the beginning of the array

```
var arr = [ 'cat', 'dog', 'goat' ];
console.log(arr) // prints 'cat', 'dog', 'goat'
arr.unshift("pug")
console.log(arr) // prints 'pug', 'cat', 'dog', 'goat'
```

shift function :

shift() method is used to remove the element from the beginning of the array, shift() functions do not accept any parameter. pop removes the element from the end of the array

```
var arr = [ 'cat', 'dog', 'goat' ];
console.log(arr) // prints 'cat', 'dog', 'goat'
arr.shift()
console.log(arr) // prints 'dog', 'goat'
```

sort() function in nodejs:

sort method is used to arrange the elements of the array in ascending order

```
var arr = [ 'cat', 'goat', 'dog', 'mummy', 'goat' ];
console.log(arr);
console.log(arr.sort());
```

output:

```
[ 'cat', 'goat', 'dog', 'mummy', 'goat' ]
```

```
[ 'cat', 'dog', 'goat', 'goat', 'mummy' ]
```

reverse() function in nodejs:

reverse() function is used to reverse the order of the array such that the first element becomes the last and the last element becomes the first. reverse() function does not accept any parameter.

```
var arr = [ 1, 9, 3 ];
console.log(arr) // prints 1, 9, 3
arr.reverse()
console.log(arr) // prints 3, 9, 1
```


concat()

concat method is used to join two arrays and returns a new array consisting of the elements of both the arrays one after another.

```
arr = ['jack fruit','grape'];
arr2 = ['mango','kiwi','apple'];
console.log(arr);
console.log(arr2);
var new_arr = arr.concat(arr2);
console.log(new_arr);
//prints 'jack fruit','grape','mango','kiwi','apple'
```

```
arr = ['jack fruit','grape'];
arr2 = ['mango','kiwi','apple'];
console.log(arr);
console.log(arr2);
var new_arr = arr2.concat(arr);
console.log("concatinated array is ", new_arr);
console.log("concatinated array after soting is" , new_arr.sort());
```

output:

```
[ 'jack fruit', 'grape' ]
[ 'mango', 'kiwi', 'apple' ]
concatinated array is [ 'mango', 'kiwi', 'apple', 'jack fruit', 'grape' ]
concatinated array after soting is [ 'apple', 'grape', 'jack fruit', 'kiwi', 'mango' ]
```

forEach() function :

The forEach() function works only on collections and is used to loop through each key in an array. An array is also one of the collection

```
const fruits = [];
fruits.push("banana", "apple", "peach");
fruits.forEach(function(i) {
  console.log(i);
});

// prints
banana
apple
```

PROGRAM 4:

Aim : Write node.js program to create, access, modify JSON Object.

JSON or JavaScript Object Notation is a light weight, text-based data interchange format.

- JSON is like XML, it is one of the way of exchanging information between applications.
- This format of data is widely used by web applications/APIs to communicate with each other.

Reading a JSON file:

- **Method 1:**

Using require method: The simplest method to read a JSON file is to require it in a node.js file using require() method.

Syntax:

```
const data = require('path/to/file/filename');
```

Example:

Create a **users.json** file in the same directory where **index.js** file present. Add following data to the json file.

users.json file:

```
[
  {
    "name": "Harber",
    "age": 30,
    "language": ["MEAN", "Express", "NodeJS"]
  },
  {
    "name": "Alex Young",
    "age": 35,
    "language": ["Angular", "MEAN", "AngularJS"]
  },
  {
    "name": "John",
    "age": 21,
    "language": ["JavaScript", "PHP", "Python"]
  },
  {
```

```
"name": "Smith",  
"age": 25,  
"language": ["PHP", "Go", "JavaScript"]  
}  
]
```

index.js file:

```
const users = require("./users");  
console.log(users);
```

>>>run the file using the command:

```
node index.js
```

OUTPUT:

```
PS G:\sai\JSON> node index.js  
[  
  {  
    name: 'Harber',  
    age: 30,  
    language: [ 'MEAN', 'Express', 'NodeJS' ]  
  },  
  {  
    name: 'Alex Young',  
    age: 35,  
    language: [ 'Angular', 'MEAN', 'AngularJS' ]  
  },  
  {  
    name: 'John',  
    age: 21,  
    language: [ 'JavaScript', 'PHP', 'Python' ]  
  },  
  { name: 'Smith', age: 25, language: [ 'PHP', 'Go', 'JavaScript' ] }  
]  
PS G:\sai\JSON> 
```

Method 2:

Using the fs module:

We can also use node.js **fs** module to read a file. The **fs** module returns a file content in string format so we need to convert it into JSON format by using **JSON.parse()** in-built method.

index.js file:

```
const fs = require("fs");

// Read users.json file
fs.readFile("users.json", function(err, data) {

    // Check for errors
    if (err) throw err;

    // Converting to JSON
    const users = JSON.parse(data);

    console.log(users); // Print users
});
```

```
const fs = require("fs");
// Read users.json file
fs.readFile("users.json", function(err, data) {
// Check for errors
    if (err) throw err;
    // Converting to JSON
    const users = JSON.parse(data);

    console.log(users); // Print users
});
```

```
PS G:\sai\JSON> node indexs.js
[
  {
    name: 'Harber',
    age: 30,
    language: [ 'MEAN', 'Express', 'NodeJS' ]
  },
  {
    name: 'Alex Young',
    age: 35,
    language: [ 'Angular', 'MEAN', 'AngularJS' ]
  },
  {
    name: 'John',
    age: 21,
    language: [ 'JavaScript', 'PHP', 'Python' ]
  },
  { name: 'Smith', age: 25, language: [ 'PHP', 'Go', 'JavaScript' ] }
]
PS G:\sai\JSON> 
```

Writing to a JSON file:

We can write data into a JSON file by using the node.js **fs** module. We can use **writeFile** method to write data into a file.

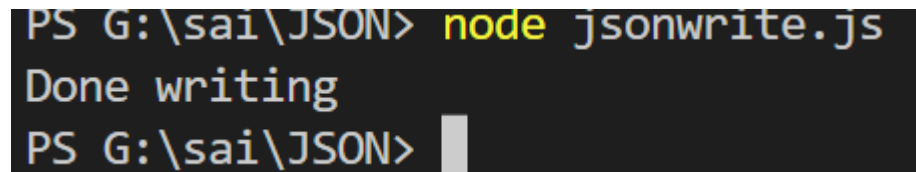
Syntax:

```
fs.writeFile("filename", data, callback);

const fs = require("fs");
// STEP 1: Reading JSON file
const users = require("./users");
// Defining new user
let user = {
  name: "New User",
  age: 30,
  language: ["PHP", "Go", "JavaScript"]
};
// STEP 2: Adding new data to users object
users.push(user);
// STEP 3: Writing to a file
fs.writeFile("users.json", JSON.stringify(users), err => {
  // Checking for errors
  if (err) throw err;

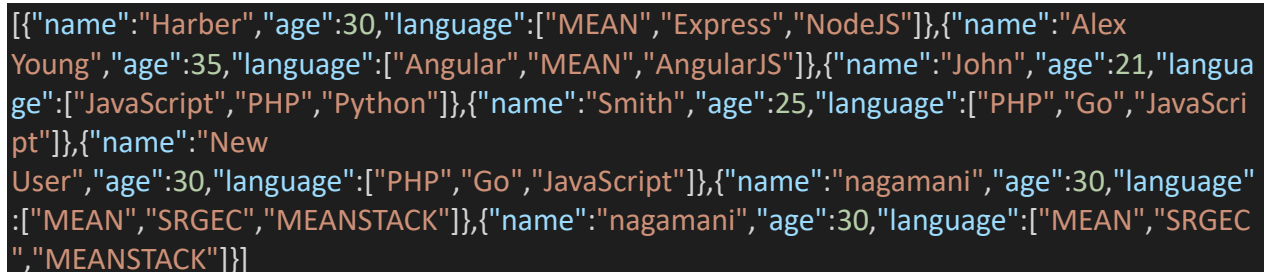
  console.log("Done writing"); // Success
});
```

OUTPUT:



```
PS G:\sai\JSON> node jsonwrite.js
Done writing
PS G:\sai\JSON>
```

>>>Now check your **users.json** file it will look something like below:



```
[{"name":"Harber","age":30,"language":["MEAN","Express","NodeJS"]}, {"name":"Alex Young","age":35,"language":["Angular","MEAN","AngularJS"]}, {"name":"John","age":21,"language":["JavaScript","PHP","Python"]}, {"name":"Smith","age":25,"language":["PHP","Go","JavaScript"]}, {"name":"New User","age":30,"language":["PHP","Go","JavaScript"]}, {"name":"nagamani","age":30,"language":["MEAN","SRGEC","MEANSTACK"]}, {"name":"nagamani","age":30,"language":["MEAN","SRGEC","MEANSTACK"]}]
```