Name of Student : Ganesh Anil Deulkar			
Roll Number : 09		**LAB Practical Number:** 02	
Title of LAB Assignment : Implementation of node js modules			
DOP : 04/09/2023		**DOS :** 16/09/2023	
CO Mapped : CO-1	**PO Mapped:** PO-3, PO-5	**Faculty Signature:**	**Marks :**

Aim :- Implementation of node.js modules.

Description :-

Node.js modules are encapsulated blocks of code that help organize and structure JavaScript applications. They enable code reuse, separation of concerns, and maintainability. Modules can be created with the **`module.exports`** object and imported using **`require()`** in other parts of the application. This modular approach enhances code readability, facilitates collaboration, and simplifies the development and maintenance of Node.js applications by breaking them into manageable, self-contained components.

Types of modules in Node.js

In Node.js, there are three primary types of modules: Core Modules, Local Modules, and Third-Party Modules.

1.) Core Modules:

Core modules are built-in modules that come bundled with Node.js. These modules provide essential functionalities like file system operations (`fs`), HTTP server handling (`http`), and more. They are accessible without the need for additional installation and serve as a foundational part of Node.js, making it possible to interact with the underlying system and network.
Some examples :-

`fs.readFile()`: Reads the content of a file asynchronously.
`http.createServer()`: Creates an HTTP server instance.
`path.join()`: Joins path segments to create a file path.
`os.totalmem()`: Returns the total system memory in bytes.
`events.EventEmitter`: Provides event-driven functionality for custom objects.
`util.inherits()`: Inherits methods and properties from one constructor to another.
`url.parse()`: Parses a URL string into its components.

2.) Custom modules :
in Node.js are user-defined modules created to encapsulate code for better organization and reusability. These modules are specific to a project and stored in separate files. To use a custom module, you export functions, objects, or variables using `module.exports`, making them accessible elsewhere in your codebase with `require()`.

Common functions in custom modules:

1. **calculateTotal(arr)** : Computes the sum of an array of numbers.
2. **validateEmail(email)** : Checks if an email address is valid.
3. **generateRandomString(length)** : Generates a random string of the specified length.
4. **connectToDatabase(url)** : Establishes a connection to a database.
5. **renderTemplate(template, data)** : Renders a dynamic HTML template with provided data.
6. **parseCSV(csvData)** : Parses CSV data into structured JavaScript objects.
7. **encryptPassword(password)** : Hashes and secures a user's password.
8. **sendNotification(message)** : Sends notifications through email or other channels.
9. **fetchDataFromAPI(url)** : Retrieves data from an external API.
10. **calculateDiscount(price, discountPercentage)** : Determines the discounted price based on the original price and discount percentage.

3.) Third-Party Modules :
Third-party modules, also known as npm (Node Package Manager) modules, are created by the Node.js community and external developers. These modules extend Node.js's functionality by providing various features and utilities, such as authentication (`passport`), database integration (`mongoose`), and application frameworks (`Express.js`). Developers can easily include third-party modules in their projects using npm, thereby reducing development time and effort while benefiting from the expertise of the broader Node.js community.

Common Functions in Third-Party Modules :

1. express(): Initializes an Express.js application, facilitating the creation of web servers and APIs.
2. bcrypt.hashSync(password, saltRounds): Hashes a password with salt for secure storage.
passport.authenticate(strategy, options): Handles authentication strategies and options for user login.

3. mongoose.connect(connectionString, options): Establishes a connection to a MongoDB database using Mongoose.
4. axios.get(url, config): Performs an HTTP GET request to a specified URL and configuration.
5. jsonwebtoken.sign(payload, secretKey, options): Generates JSON Web Tokens (JWTs) for secure authentication.
6. body-parser.json(): Middleware to parse JSON data from HTTP requests in Express.js.
7. validator.isEmail(email): Validates if a string is a valid email address.
8. multer({ storage }): Middleware for handling file uploads in Express.js with customizable storage options.
9. cors(options): Middleware to enable Cross-Origin Resource Sharing (CORS) for secure API requests.

Built in Module :

i) Write a program to print information about the computer's operating system using the OS module(use any 5 methods).

Code :-

```
var path = require('path');
var os = require('os');
const { uptime } = require('process');
var filename = path.basename('"D:\k1\require.js"');
var dname = path.dirname('"D:\k1\require.js"');
var ename = path.extname('"D:\k1\require.js"');

console.log(filename);
console.log(dname);
console.log(ename);
console.log("Architecture=", os.arch());
console.log("Endianness=", os.endianness());
console.log("freemem=", os.freemem());
console.log("hostname=", os.hostname());
console.log("Networkinterfaces=-", os.networkInterfaces());
console.log("platform=", os.platform());
console.log("Release=", os.release());
console.log("Tmpdir=", os.tmpdir());
console.log("Totalmem=", os.totalmem());
console.log("Type=", os.type());
console.log("Uptime=", os.uptime());
console.log("Useinfo=", os.userInfo());
```

Output :-

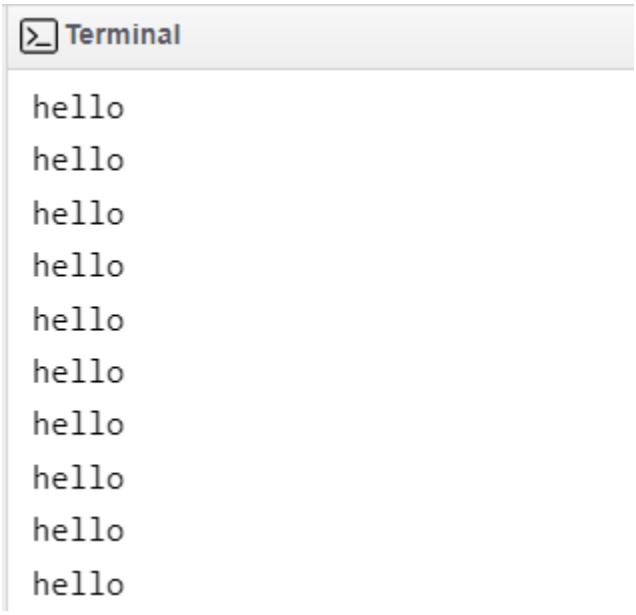
```
PS C:\Users\Exam> node D:\k1\require.js
equire.js"
.
.js"
Architecture= x64
Endianness= LE
freemem= 2153631744
hostname= MCAB14-33
Networkinterfaces=- {
  Ethernet: [
    {
      address: 'fe80::2731:e9df:1ca7:37e',
      netmask: 'ffff:ffff:ffff:ffff::',
      family: 'IPv6',
      mac: 'f4:6b:8c:8d:12:4b',
      internal: false,
      cidr: 'fe80::2731:e9df:1ca7:37e/64',
      scopeid: 5
    },
    {
      address: '192.168.42.193',
      netmask: '255.255.240.0',
      family: 'IPv4',
      mac: 'f4:6b:8c:8d:12:4b',
      internal: false,
      cidr: '192.168.42.193/20'
    }
  ],
  'Loopback Pseudo-Interface 1': [
    {
      address: '::1',
      netmask: 'ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff',
      family: 'IPv6',
      mac: '00:00:00:00:00:00',
      internal: true,
      cidr: '::1/128',
```

ii) Print "Hello" every 500 milliseconds using the Timer Module. The message should be printed exactly 10 times. Use SetInterval ,ClearInterval and SetTimeout methods.

Code :-

```
var myint = setInterval(function(){
       console.log("hello");},500)
var stop =setTimeout(function() {
 var p = clearInterval(myint)},5500);
```

Output :-


Terminal
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello

Custom Module :

i) create a Calculator Node.js Module with functions add, subtract and multiply,Divide. And use the Calculator module in another Node.js file.

Code :-

First creating a .js module with functions :

```
exports.add = function (a, b) {
   return (a + b);
}
exports.sub = function (a, b) {
   return (a - b);
}
exports.mul = function (a, b) {
   return (a *b);
}
exports.div = function (a, b) {
   return (a /b);
}
exports.mod = function (a, b) {
   return (a%b);
}
```

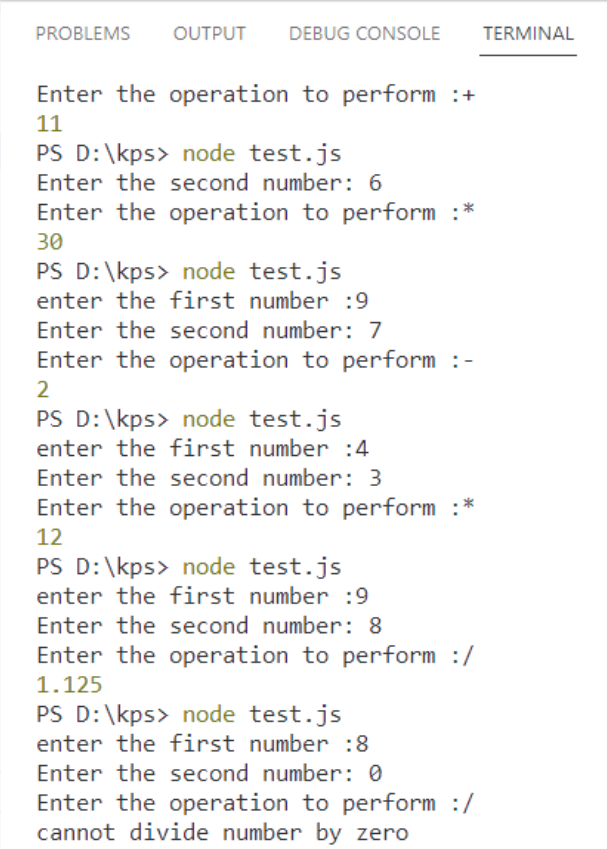
Now creating a separate file to import the above functions :

```
var file = require('./custom.js');
const prompt = require('prompt-sync')();
var a = parseInt(prompt("enter the first number :"));
var b = parseInt(prompt("Enter the second number: "));
var output = prompt("Enter the operation to perform :");
switch (output) {
   case "+":
```

```
        console.log(file.add(a,b));
        break;

    case "-":
        console.log(file.sub(a, b));
        break;
    case "*":
        console.log(file.mul(a, b));
        break;
    case "/":
        if (b == 0) {
            console.log("cannot divide number by zero");
        }
        console.log(file.div(a, b));
        break;
    case "%":
        console.log(file.mod(a, b));
        break;
}
}
```

Output :-



ii) Create a circle module with functions to find the area and perimeter of a circle and use it.

1.) Creating a file for exporting the function :

```
exports.perimeter = function( r){
    return (2*3.14*r);
}

exports.area = function (a) {
    return(3.14*a*a)
}
```

2.) Creating a different .js file to take the input from the user and export the above function :

```
var file = require('./custom.js');
const prompt = require('prompt-sync')();
var num = parseInt(prompt("Radius of the circle:"));
console.log(“Area of the circle: $(file.area(num)) );
console.log(“Perimeter of the circle: $(file.perimeter(num)) );
```

Output :-

```
Radius of the circle: 5
Area of the circle: 78.53981633974483
Perimeter of the circle: 31.41592653589793
```

Conclusion :-

The Different Modules in Node.js, their different types and their most common functions have been studied with the help of the above practical. Moreover the various problems related to the Built in module and Custom modules have been solved by writing and executing the programs.