**Aim: Write a program to demonstrate built-in module in node.js**

**Objective: To understand the working of built-in modules**

**Scope:** Application of built-in modules in different ways.

**Theory:-** Node.js has a set of built-in modules which you can use without any further installation.

# Node.js Module

Module in Node.js is a simple or complex functionality organized in single or multiple JavaScript files which can be reused throughout the Node.js application.

Each module in Node.js has its own context, so it cannot interfere with other modules or pollute global scope. Also, each module can be placed in a separate .js file under a separate folder.

Modules are encapsulated code blocks that communicate with an external application. These can be a single file or a collection of multiple files/folders. These are reusable, hence they are widely used.

### Types of Modules in Nodejs:

There are three types of modules  
1) Core Modules  
2) local Modules  
3) Third-party Modules

#### 1.Nodejs Core Modules:

Built-in modules of node.js that are part of nodejs and come with the Node.js installation process are known as core modules.

To load/include this module in our program, we use the **require** function.

let module = require('module\_name')

The return type of require() function depends on what the particular module returns.

Http, file system and url modules are some of the core modules.

#### 2. Nodejs Local Modules:

Local modules are created by us locally in our Node.js application. These modules are included in our program in the same way as we include the built in module.

**Code for creating local modules and exporting:**

exports.add=**function**(n,m){

**return** n+m;};

Exports keyword is used to make properties and methods available outside the file.

In order to include the add function in our index.js file we use the require function.

**Code for including local modules:**

let sum = require('./sum')

console.log("Sum of 10 and 20 is ", sum.add(10, 20))

Add the above code in a index.js file

To run this file, open a terminal in the project directory and type node index.js and press enter. You can see the result of addition of 10 and 20. This addition has been performed by the add function in the sum module.

#### 3. Nodejs Third Party Modules:

Modules that are available online and are installed using the npm are called third party modules.

Examples of third party modules are express, mongoose, etc.

To install third party modules refer to the previous blog where we have discussed how to install modules using npm.

**Custom Module**

Modules are the collection of JavaScript codes in a separate logical file that can be used in external applications on the basis of their related functionality. Modules are popular as they are easy to use and are reusable.

To create a module in Node.js, will need the **exports** keyword. This keyword tells Node.js that the function can be used outside the module.

**Syntax:**

**exports**.function\_name = **function**(arg1, arg2, ....argN) {

// function body

};

|  |
| --- |
| Create a file to export // File name: calc.js  exports.add = function (x, y) {      return x + y;  };    exports.sub = function (x, y) {      return x - y;  };    exports.mult = function (x, y) {      return x \* y;  };    exports.div = function (x, y) {      return x / y;  }; |

Use the ‘require’ keyword to import the file

|  |
| --- |
| // File name: App.js  var calculator = require('./calc');    var x = 50, y = 20;    console.log("Addition of 50 and 20 is "                     + calculator.add(x, y));    console.log("Subtraction of 50 and 20 is "                     + calculator.sub(x, y));  console.log("Multiplication of 50 and 20 is "                     + calculator.mult(x, y));    console.log("Division of 50 and 20 is "                     + calculator.div(x, y)); |

## Node.js Core Modules/Built-in Modules

The following table lists some of the important core modules in Node.js.

| Core Module | Description |
| --- | --- |
| [http](https://nodejs.org/api/http.html) | http module includes classes, methods and events to create Node.js http server. |
| [url](https://nodejs.org/api/url.html) | url module includes methods for URL resolution and parsing. |
| [querystring](https://nodejs.org/api/querystring.html) | querystring module includes methods to deal with query string. |
| [path](https://nodejs.org/api/path.html) | path module includes methods to deal with file paths. |
| [fs](https://nodejs.org/api/fs.html) | fs module includes classes, methods, and events to work with file I/O. |
| [util](https://nodejs.org/api/util.html) | util module includes utility functions useful for programmers. |

### Loading Core Modules

In order to use Node.js core or NPM modules, you first need to import it using require() function as shown below.

var module = require('module\_name');

As per above syntax, specify the module name in the require() function. The require() function will return an object, function, property or any other JavaScript type, depending on what the specified module returns.

The following example demonstrates how to use Node.js http module to create a web server.

Example: Load and Use Core http Module

var http = require('http');

var server = http.createServer(function(req, res){

//write code here

});

server.listen(5000);

In the above example, require() function returns an object because http module returns its functionality as an object, use its properties and methods using dot notation e.g. http.createServer().

Conclusion: Different built-in modules are studied in Node.js successfully.