Node.js

#### **Introduction**

* Node.js is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside a web browser. In other words, it is an open-source environment.
* Node.js uses asynchronous programming.
* Node.js runs single-threaded, non-blocking, asynchronously programming.
* It is very memory efficient.

A common task for a web server can be to open a file on the server and return the content in the file to the client. PHP or ASP handles such file requests in the following way:

1. Sends the task to the computer's file system.
2. Waits while the file system opens and reads the file.
3. Returns the content to the client.
4. Ready to handle the next request.

Node.js handles the same situation in a different way. It performs the following actions:

1. Sends the task to the computer's file system.
2. Ready to handle the next request.
3. When the file system has opened and read the file, the server returns the content to the client.

You can conclude that Node.js eliminates the waiting time, unlike PHP or ASP. It continues with the next request

**Tasks performed**

* It can generate dynamic page content.
* It can create, open, read, write, delete, and close files on the server.
* It can collect form data.
* It can add, delete, modify data in your database.

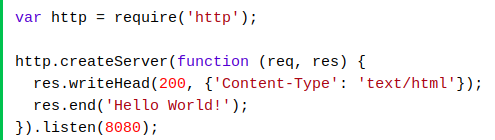
**Node.js file**

* Node.js files contain tasks that will be executed on certain events. For example, an event where someone is trying to access a port on the server.
* These files must be initiated on the server before using them.
* Node.js files have extension ‘.js’.

**Getting started with Node.js**

The official Node.js website has installation instructions for Node.js: [https://nodejs.org](https://nodejs.org/).

Once you have downloaded and installed Node.js on your computer, let's try to display ‘Hello World’ in a web browser. This can be performed as follows:

1. Create a Node.js file named "myfirst.js"
2. Add the provided code:  
   
3. Save the file on your computer as **C:\Users\Your Name\myfirst.js**.  
   **Note**: This code instructs the computer to write ‘Hello World!; if anyone (for example, a web browser) is trying to access your computer on port 8080.

**Command Line Interface**

To access a Node.js file, it must be initiated in the command line interface program of your computer.

**Note**:

* The process of opening the command line interface depends on the type of operating system you are using.
* The following instructions are provided for Windows users only.

**Steps to open the command line interface**

1. Press the start button.
2. Click on the Command Prompt option or you can write **cmd** in the search field provided.
3. Navigate to the folder that contains your file named ‘myfirst.js’.
4. The command line interface window will be displayed as follows:  
   

## **Initiate the Node.js file**

The file you have created must be initiated by Node.js before you start using it. To perform this, you are required to follow these steps:

1. Start your command line interface.
2. Write node myfirst.js and then hit the enter button on your keyboard. The following information is displayed:  
   

Now, your computer works as a server!

If anyone tries to access your computer on port 8080, they ‘Hello World!’ message is displayed.

**Node.js modules**

You can consider modules to be the same as the JavaScript libraries. They represent a set of functions that you want to include in your application.

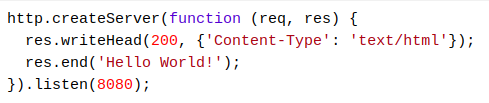
## **Built-in modules**

There are some set of built-in modules that you can use without any further installations. You can refer to <https://www.w3schools.com/nodejs/ref_modules.asp> to know more about the built-in modules of Node.js version 6.10.3.

If you want to include a module, use the require() function with the name of the module that you want to include. For example, you want to include a module called HTTP. Then you must perform as follows:

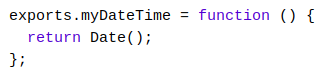


Now your application has access to the HTTP module. This has enabled it to create a server as shown in the following code:



**User-defined modules**

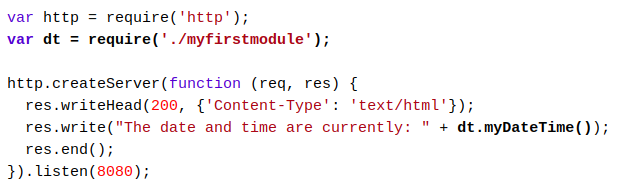
You can create your own modules and easily include them in your applications. The following example creates a module that returns a date and time object:



**Note**: You can use the **exports** keyword to allow properties and methods to be available outside the module file.

Don’t forget to save the code above in a file called ‘**myfirstmodule.js**’!

You can now include your own module in any of your Node.js files. Consider the following example:



**Note**: You can use ./ to locate the module. This indicates that the module is located in the same folder as the Node.js file.

Now, save the code above in a file called ‘**demo\_module.js**’. You can initiate the file as follows:



# **Node.js HTTP Module**

**The Built-in HTTP Module**

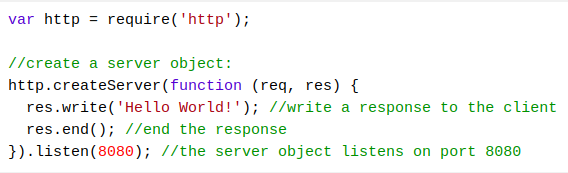
Node.js has a built-in module called HTTP. It allows Node.js to transfer data over the HTTP protocol.

If you want to include the HTTP module, you are required to use the require() method. It can be performed as follows:



**Node.js as a Web Server**

The HTTP module can create an HTTP server that listens to the server ports and provides a response back to the client. To perform this, you can use the createServer() method as follows:



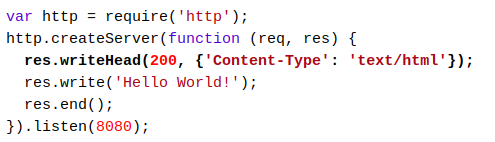
The function that is passed into the **http.createServer()** method is only executed when someone tries to access the computer on port 8080.

Now, you are required to save the code written in a file. Let’s call it ‘demo\_http.js’. Further, you can initiate the file as follows:



## **Add an HTTP Header**

If you want the HTTP server to display the response in HTML, then you must include an HTTP header with a correct content type. An example is shown as follows:

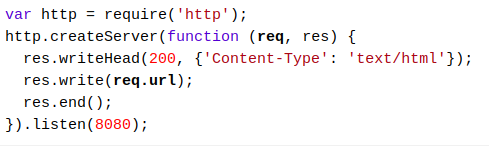


**Note**: In the example provided, the first argument (200) of the res.writeHead() method represents that all the functionalities are working correctly. The second argument represents an object that contains the response header.

## **Read the Query String**

The function passed into the http.createServer() has a req argument that represents the request from the client as an object (http.IncomingMessage object).

This object has a property called URL that holds the part of the URL that comes after the domain name. It is displayed as follows:

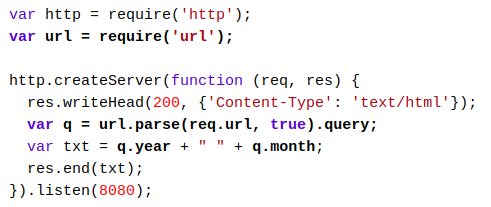


Now, save the code in a file called ‘demo\_http\_url.js’. Again initiate the file as follows:



## **Split the Query String**

There are built-in modules to easily split the query string into readable parts such as a URL module. The following example splits the query string into readable parts:



Again, save the provided code in a file called ‘demo\_querystring.js’. Then, initiate the file as follows:  


# **Node.js File System Module**

The Node.js file system module allows you to work with the file system on your computer. To include the File System module you must use the **require()** method as follows:

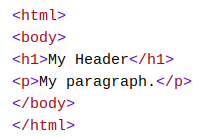


A common use for the File System module:

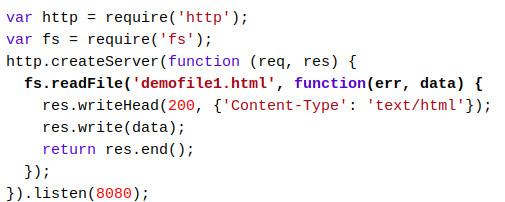
* Read files
* Create files
* Update files
* Delete files
* Rename files

## **Read files**

The fs.readFile() method is used to read files on your computer. Assume that you have the following HTML file (located in the same folder as Node.js):



You can create a Node.js file that reads the HTML file and returns the content. It is performed as follows:



You can save the provided code in a file called ‘**demo\_readfile.js**’ and initiate the file in the following way:



## **Create Files**

The File System module has various methods for creating new files. They are as follows:

* fs.appendFile()
* fs.open()
* fs.writeFile()

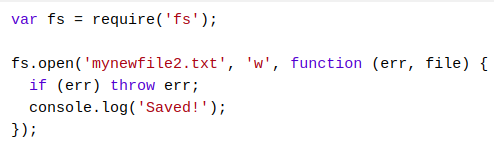
The fs.appendFile() method appends a specified content to a file. If the file does not exist, a new file will be created.

For example, you can create a new file using the appendFile() method.



* The fs.open() method takes a ‘flag’ as the second argument. Here, if the flag is ‘w’ that represents ‘writing’, then the specified file is opened for writing.
* If the file does not exist, then an empty file is created.

### You can create a new and empty file using the open() method. It can be performed as follows:

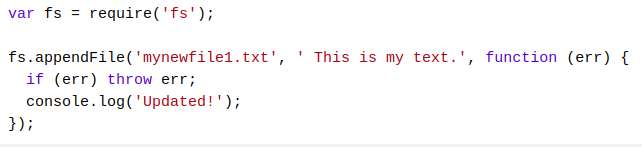


## **Update Files**

The File System module has various methods for updating files. They are as follows:

* fs.appendFile()
* fs.writeFile()

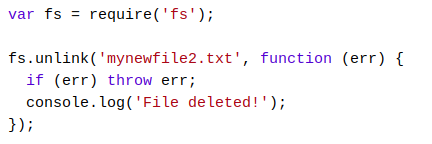
The fs.appendFile() method appends the specified content at the end of the specified file. An example is illustrated where you want to append ‘This is my text’ to the end of the file.



## **Delete Files**

If you want to delete a file with the File System module, then you can use the fs.unlink() method.

**Note**: The fs.unlink() method deletes the specified file as shown in the following example:



## **Rename Files**

If you want to rename a file with the File System module, then you can use the fs.rename() method.

**Note**: The fs.rename() method renames the specified file as shown in the following example:



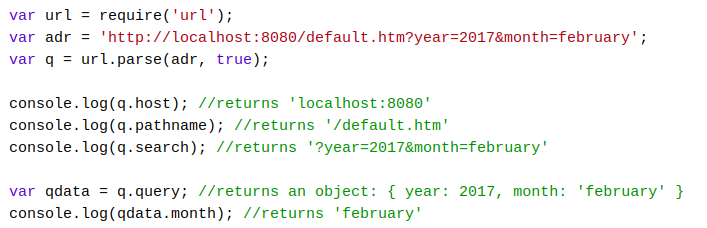
# **Node.js URL Module**

## **The Built-in URL Module**

The URL module splits up a web address into readable parts. If you want to include the URL module, then use the require() method as follows:



Parse an address with the url.parse() method. This will return a URL object with each part of the address as properties. The following example illustrates this scenario:

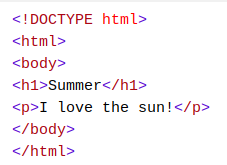


## **Node.js File Server**

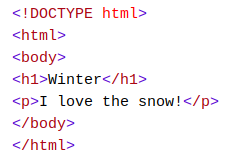
## Now you know how to parse the query string and how to make Node.js behave as a file server. Let’s combine the two concepts and serve the file requested by the client.

First, create two HTML files and save them in the same folder as your node.js files.

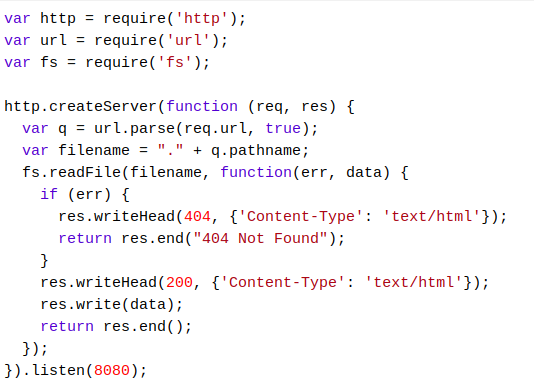
**File 1**



**File 2**



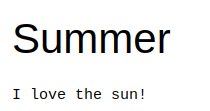
Now, create a Node.js file that opens the requested file and returns the content to the client. If anything goes wrong, then you can throw a 404 error. An example is provided.



Then, initiate the file as follows:



The result for file 1 is as follows:



The result for file 2 is as follows:



**Node.js NPM**

* NPM is a package manager for Node.js packages or modules.
* The NPM program is installed on your computer when you install Node.js.
* A package in Node.js contains all the files you require for a module.
* Modules are JavaScript libraries that you can include in your project.

**Download a Package**

Open the command line interface.

Instruct NPM to download the package that you require. For example, you want to download a package called ‘upper case’, then you are required to instruct the NPM in the following way:

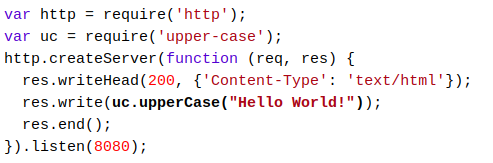


**Note**: NPM creates a folder named ‘node\_modules’ where the package can be placed. All packages you install in the future are stored in this folder.

**Using a Package**

Once the package is installed, it is ready to use.

The following example illustrates how to create a Node.js file that can convert the output ‘Hello World!’ into upper-case letters:

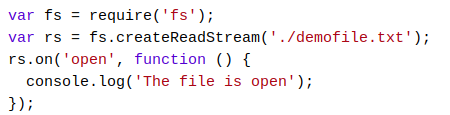


Save the code above in a file called ‘demo\_uppercase.js’ and initiate the file as follows:



**Events in Node.js**

Every action on a computer is an event. Objects in Node.js can fire events. For example, the readStream object fires events when opening and closing a file as follows:



**Events Module**

Node.js has a built-in module called ‘Events’. Here, you can create-, fire-, and listen for- your own events.

To include the built-in Events module you can use the require() method. In addition, all event properties and methods are an instance of an EventEmitter object. To be able to access these properties and methods you can create an EventEmitter object as follows:

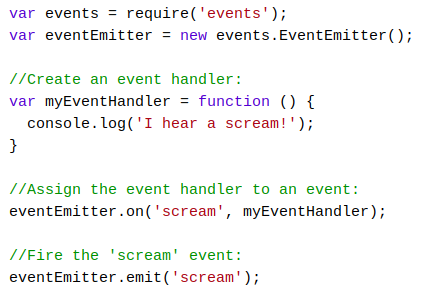
****

## **The EventEmitter Object**

You can assign event handlers to your own events with the EventEmitter object.

The following example creates a function that can execute when a ‘scream’ event is fired.

**Note**: To fire an event you can use the emit() method.



## **The Formidable Module**

The Formidable module can be downloaded and installed using NPM as follows:



After you have downloaded the Formidable module, you can include the module in any application by using the following command:



## Upload Files

Now you are ready to make a web page in Node.js that lets the user upload files to your computer:

## Step 1: Create an Upload Form

Create a Node.js file that writes an HTML form with an upload field. The following example illustrates a code to produce an HTML form:



## Step 2: Parse the Uploaded File

## You must include the Formidable module such that it can parse the uploaded file once it reaches the server.

## When the file is uploaded and parsed, it is stored on a temporary folder on your computer.

The following example shows how a file can be uploaded and placed on a temporary folder:



## Step 3: Save the File

## When a file is successfully uploaded to the server, it is placed on a temporary folder.

## The path to this directory can be found in the ‘files’ object which is passed as the third argument in the parse() method's callback function.

## To move the file to the folder of your choice, use the File System module, and rename the file as follows:



# **Node.js Send an Email**

## **The Nodemailer Module**

* The Nodemailer module makes it easy to send emails from your computer.
* The Nodemailer module can be downloaded and installed using npm as follows:  
  

After you have downloaded the Nodemailer module, you can include the module in any application using the following command:



## **Send an Email**

Now, you are ready to send emails from your server. You can use the username and password from your selected email provider to send an email.

**Note**: This tutorial will show you how to use your Gmail account to send an email.



## **Multiple Receivers**

## To send an email to more than one receiver, add them to the ‘to’ property of the mailOptions object. The receivers must be comma-separated. The following example sends an email to more than one address:

## 

## **Send HTML**

## To send HTML formatted text in your email, you are required to use the ‘html’ property instead of the ‘text’ property. The following code illustrates this scenario:



-------reference credits: w3schools, tutorialspoint,<https://github.com/hegdenaveen1/Become-A-Full-Stack-Web-Developer#start-here> ------