

## BTech:IIIYear

Departmentof Computer Science &Information Technology

**Name of the Student : UJJAWAL SAHU**

**Branch & section : CSIT -3**

**Roll No. : 0827CI201193**

**Year : 2020-24**

## Department of Computer Science & Information Technology

**AITR,Indore**

**Introduction**

Node.js is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside of a web browser. Node.js is a popular, lightweight web framework for beginners, and it is used by many big companies like Netflix and Uber.

When we typically think of JavaScript, our mind tends to go to web development. Until Node.js came along, there was really no way to run JavaScript outside of a browser. When we write a backend server and database, Node.js is a popular choice because we can run our code as a standalone application rather than something that can only be evaluated in a browser environment.

## **Fundamentals of Node.js**

Now that we know what Node.js is, let’s explore the fundamentals of this tool..

1. **Console**
2. **Buffer**
3. **File System**
4. **Event Loop**
5. **Globals**
6. **Console:**-

The **console** is a module provided by Node.js that is akin to the JavaScript console in the browser when you inspect a webpage. The console has methods that are available for us to use for debugging purposes.

* console.log(): Frequently used to log some sort of output.
* console.warn(): Explicitly delivers a warning to the console.
* console.error(): Explicitly delivers an error message to the console. You can log an error as a string or as an object. If logged as a new Error(), a traceback will be included as part of the message.
* console.trace(): Logs a traceback when an error occurs in your code. Gives line number and column number of the file that the error probably occurred.

### **Buffer**

At its core, the **Buffer class** in Node.js is a temporary storage solution for file systems. Due to its low-level nature, as web developers we will rarely actually use the Buffer class directly. The main purpose of this class is to allocate memory.

Let’s take a look at a few methods that the Buffer class provides.

### **File System**

The **file system (fs) module** allows us to interact with files in Node.js. There are synchronous and asynchronous methods that can be used to read or write to a file using the fs module. In contrast to using console or the Buffer class, we need to import the fs module into the file that we would like to use in order to get it to work.

### **Event Loop**

Much of Node.js is built to be **event-driven**. When a user clicks on an interface or types in a form, an event is triggered to happen and then something occurs as a result. To attach a function or set of functions to a specific event is emitting an event.

These functions, called event listeners, are one part of an overall journey called the Event Loop

### **Globals**

Global objects are available in every module, so they can be used without importing a specific module. The Buffer class, for example, class is defined as a global in Node.js. Some other common global objects are:

* The console object is used to print to stdout and stderr.
* Timers, such as setImmediate, setInterval, and setTimeout, are also globals.
* The process object is also global.

## **How to build a basic Node.js project**

Let’s learn how to get started with Node.js by creating a simply Node.js [file.In](http://file.in/" \t "_blank) this example, we will be setting up our computer to work as a server!

If you would like to learn how to create a Node.js app, please see Educative beginner course.

### **Install Node.js and NPM**

First, you need to Go to the site [Node.js site](https://nodejs.org/en/download/) and download the files.

Follow the installation prompts and restart your machine for best results.

Another way you can install Node.js is to use a package manager.

Then, test that it’s working by printing the version using the following command:

> node -v

You should also test npm by printing the version using the following command:

> npm -v

### Create a file:-

Once you have installed Node.js properly, create a Node.js file. In this example, we have named it named “first.js”. We then add the following code and save the file on your computer like so: C:\Users\Your Name\first.js

var http = require('http');  
  
http.createServer(function (req, res) {  
  res.writeHead(200, {'Content-Type': 'text/html'});  
  res.end('Hello World!');  
}).listen(8080);

This code is essentially telling the computer to print “Hello World!” when accessed on port 8080.

### Command line interface:-

Node.js files must be initiated in your computer’s “Command Line Interface” program. Navigate to the folder that contains the file “first.js”.

C:\Users\Your Name>\_

### Initiate your file:-

This file needs to then be initiated by Node.js. Do this by starting your command line interface, writing node first.js, and clicking enter:

C:\Users\Your Name>node myfirst.js

Awesome! Now your computer is set up as a server, so when you accesses the computer on port 8080, the"Hello World!" message will print.

To see this in real time, open your browser, and type: http://localhost:8080

## **Conclusion:-**

Node.js lets developers use JavaScript to write command line tools and for [server-side scripting](https://en.wikipedia.org/wiki/Server-side_scripting). The functionality of running scripts server-side produces [dynamic web page](https://en.wikipedia.org/wiki/Dynamic_web_page) content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm,[[6]](https://en.wikipedia.org/wiki/Node.js#cite_note-6) unifying [web-application](https://en.wikipedia.org/wiki/Web_application) development around a single programming language, rather than different languages for server-side and client-side scripts.

**Reference-**

**sir**

**https://www.fortinet.com/**