Node.js

Node.js is a javascript runtime environment built on Chrome’s V8 Javascript engine.

What Can Node.js Do?

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

What is a Node.js File?

* Node.js files contain tasks that will be executed on certain events
* A typical event is someone trying to access a port on the server
* Node.js files must be initiated on the server before having any effect
* Node.js files have extension ".js"

What is a Module in Node.js?

Consider modules to be the same as JavaScript libraries.

A set of functions you want to include in your application.

Built-in Modules

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

Look at our [Built-in Modules Reference](https://www.w3schools.com/nodejs/ref_modules.asp) for a complete list of modules.

Include Modules

To include a module, use the require() function with the name of the module:

The Node. js Read-Eval-Print-Loop (REPL) is an interactive shell that processes Node. js expressions. The shell reads JavaScript code the user enters, evaluates the result of interpreting the line of code, prints the result to the user, and loops until the user signals to quit. The REPL is bundled with every Node.

The **node** command is the one we use to run our Node.js scripts:

node script.js

BashCopy to clipboard

If we run the **node** command without any script to execute or without any arguments, we start a REPL session:

node

BashCopy to clipboard

**Note: REPL stands for Read Evaluate Print Loop, and it is a programming language environment (basically a console window) that takes single expression as user input and returns the result back to the console after execution. The REPL session provides a convenient way to quickly test simple JavaScript code.**

> console.log('test')

Test

undefined>

The REPL has some special commands, all starting with a dot **.**. They are

* **.help**: shows the dot commands help
* **.editor**: enables editor mode, to write multiline JavaScript code with ease. Once you are in this mode, enter ctrl-D to run the code you wrote.
* **.break**: when inputting a multi-line expression, entering the .break command will abort further input. Same as pressing ctrl-C.
* **.clear**: resets the REPL context to an empty object and clears any multi-line expression currently being input.
* **.load**: loads a JavaScript file, relative to the current working directory
* **.save**: saves all you entered in the REPL session to a file (specify the filename)
* **.exit**: exits the repl (same as pressing ctrl-C two times)

Modules

const hello = {

  name: "john",

  place: "palakkad",

  great: function () {

    console.log("I am " + this.name + " From " + this.place);

  },

  message: function () {

    console.log("Good Afternoon");

  },

};

module.exports = hello;

**Node.js Global Objects**

* Node.js Global Objects are the objects that are available in all modules
* These are built in objects/functions that are part of javascript and can be used in our applications without importing any particular module

**OS Module**

* This module provides many functions that you can use to retrieve information from the underlying operating system and the computer the program runs on ,and interact with it

var os=require('os');

console.log(os.type());

console.log(os.arch());

console.log(os.platform());

console.log(os.version());

console.log(os.release());

console.log(os.uptime());

console.log(os.userInfo());

console.log(os.totalmem());

console.log(os.freemem());

console.log(os.cpus());

console.log(os.networkInterfaces());

**File System Module**

* TheNode.js file system module allows you to work with the file system on your computer
* Common uses

Read Files

Create files

Update Files

Delete files

Rename FIles

**Read Files**

var fs = require("fs");

fs.readFile("./text.txt", "utf8", function (err, data) {

  if (err) {

    console.error(err);

  }

  console.log(data);

});

**Write Files**

var fs = require("fs");

fs.WriteFile("new1.txt","this is a example file 2 ",(err)=>{

    if(err){

        console.log(err)

    }

    console.log("file created");

    }

);

**Append File**

var fs = require("fs");

fs.appendFile("new1.txt","this is a example file 2 ",(err)=>{

    if(err){

        console.log(err)

    }

    console.log("file created");

    }

);

**Delete File**

var fs = require("fs");

  fs.unlink("./text.txt",(err)=>{

    if(err){

        console.log(err)

    }

    console.log("file deleted");

    }

);

**PathModule**

* The path module allows you to interact with file paths easily
* The path modules has many useful properties and methods to access and manipulate paths in the file sytem+

var path=require("path");

const p="D:\Gtec\Node\_js\file.pdf";

console.log(path.basename(p));

console.log(path.dirname(p));

console.log(path.extname(p));

console.log(path.join("/Gtec","node\_js","file.js"));

console.log(path.parse(p));

**HTTP Modules**

* HTTP module allows Node.js to transfer data over the Hyper Text Transfer Protocol(HTTP)
* The HTTP module can create an HTTP server that listens to server ports and gives a response to the client

const http =require("http");

const url =require("url");

const server =http.createServer((req,res)=> {

    const rUrl =url.parse(req.url,true).pathname;

    if (rUrl == "/"){

        res.write("Home Page");

        res.end();

    }

   else if (rUrl == "/about"){

        res.write("About Page");

        res.end();

    }

    else if (rUrl == "/contact"){

        res.write("Contact Page");

        res.end();

    }

});

server.listen(8080);

const si=setInterval(function () {

  console.log("Node.js");

}, 1000);

const st=setTimeout (() => {

  clearInterval(si);

}, 4000);

clearTimeout(st);