Node.js

Main Features

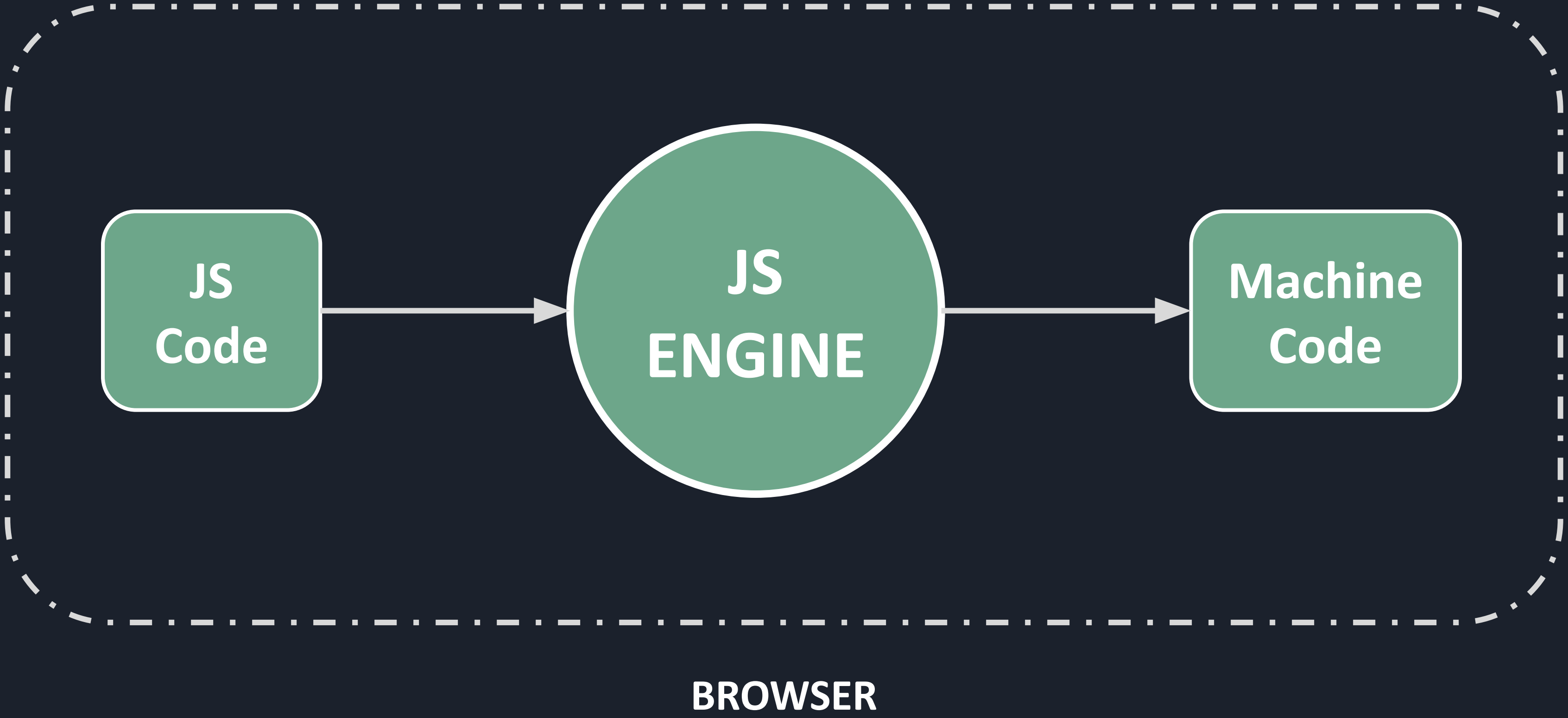
- Runs in a single process, without creating a new thread for every request.
- Runs on V8 JavaScript engine, the core of Google Chrome, outside of the browser.
- Provides a set of asynchronous I/O primitives in its standard library that prevent JavaScript code from blocking.
- Instead of blocking the thread and wasting CPU cycles waiting, Node.js will resume the operations when the response comes back.

Node.js

og mmi
Lang e

A **runtime environment** for executing JavaScript code

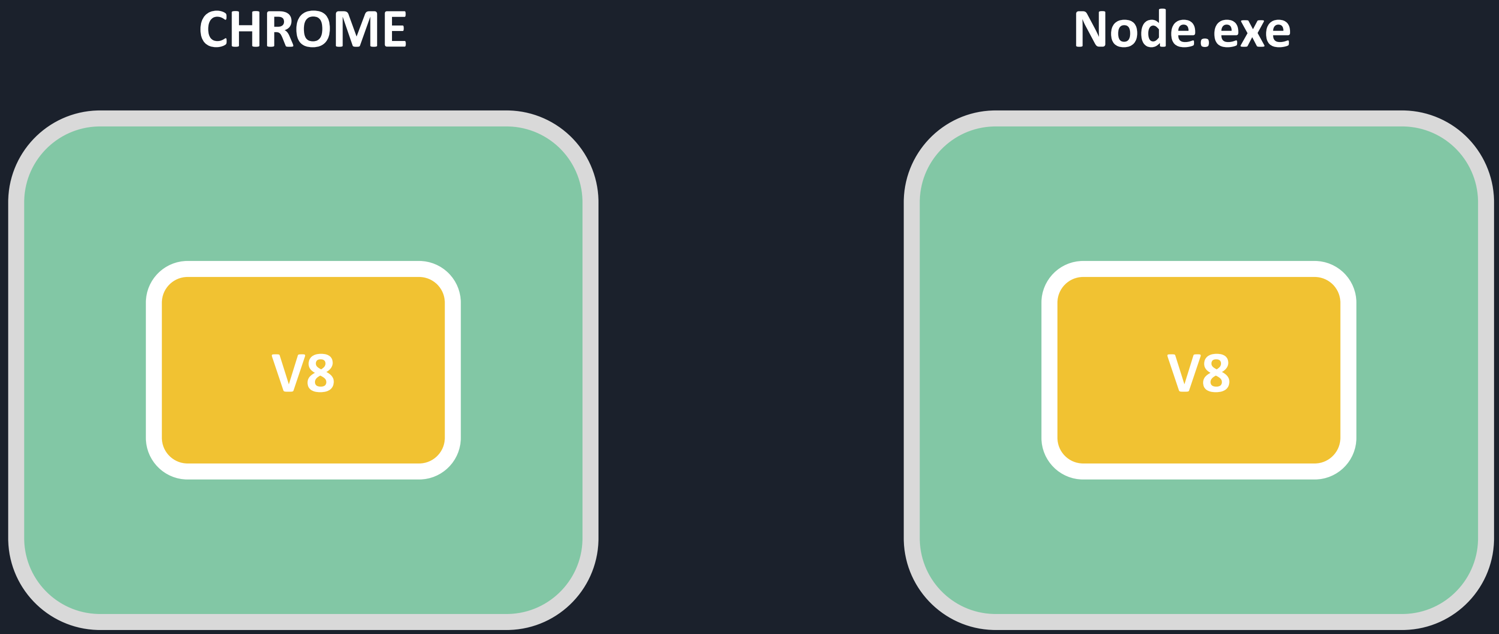
What is a runtime environment?



Chakra

SpiderMonkey

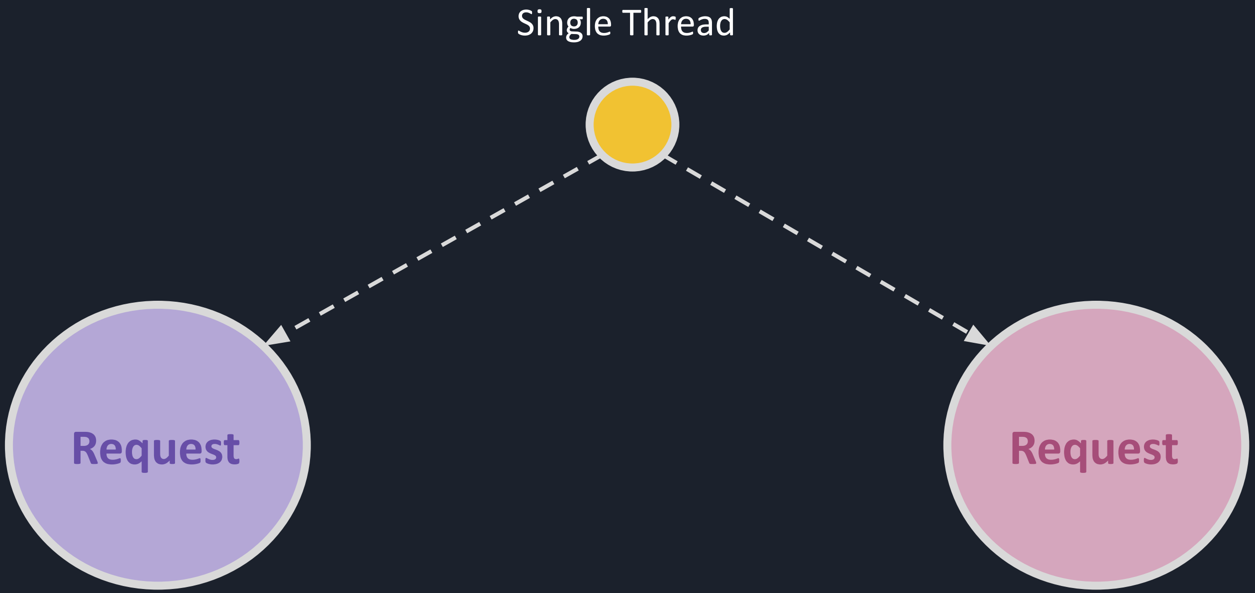
v8


CHROME
V8
Node.exe
V8

How Node.js Works?

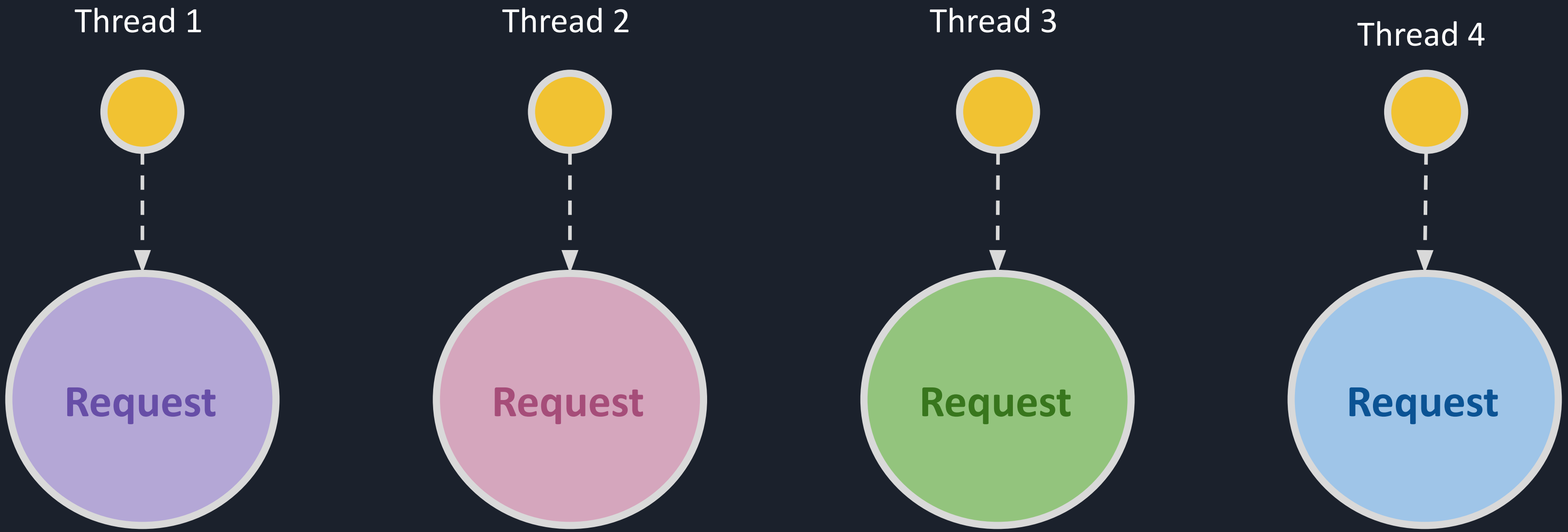
Non-blocking

ASYNCHRONOUS

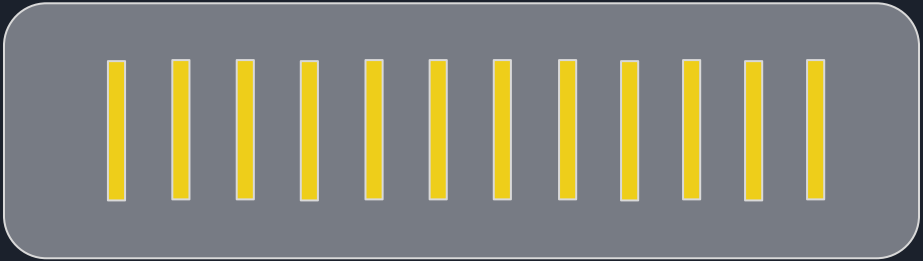


Blocking

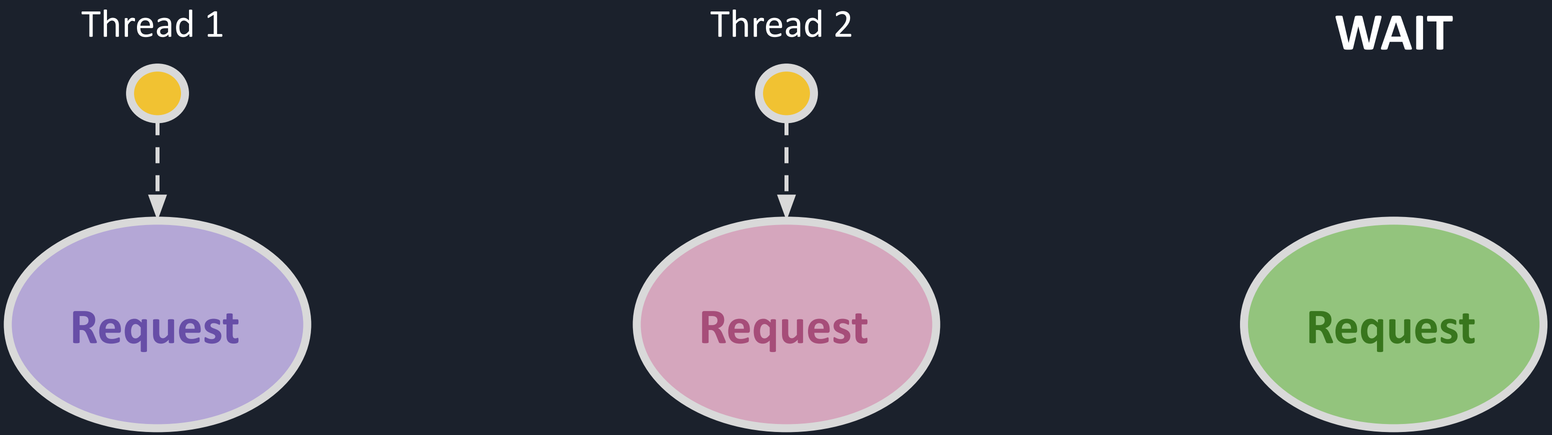
SYNCHRONOUS

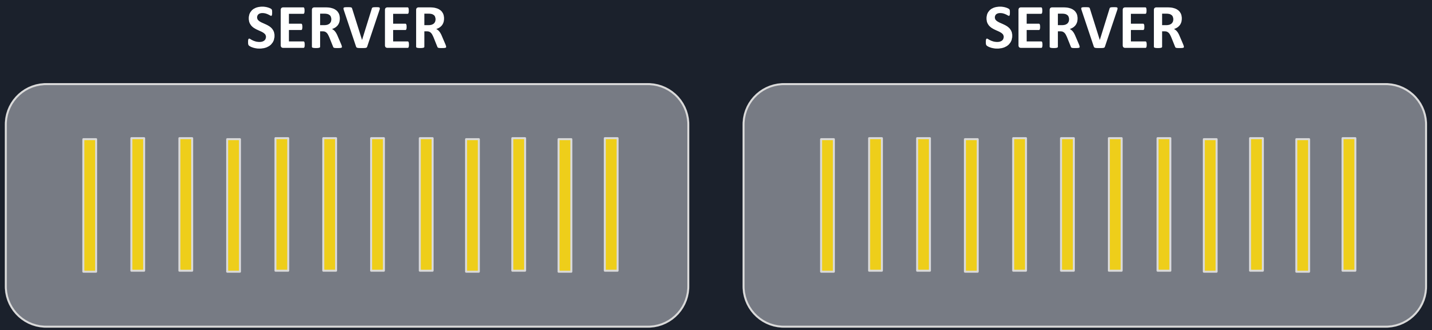


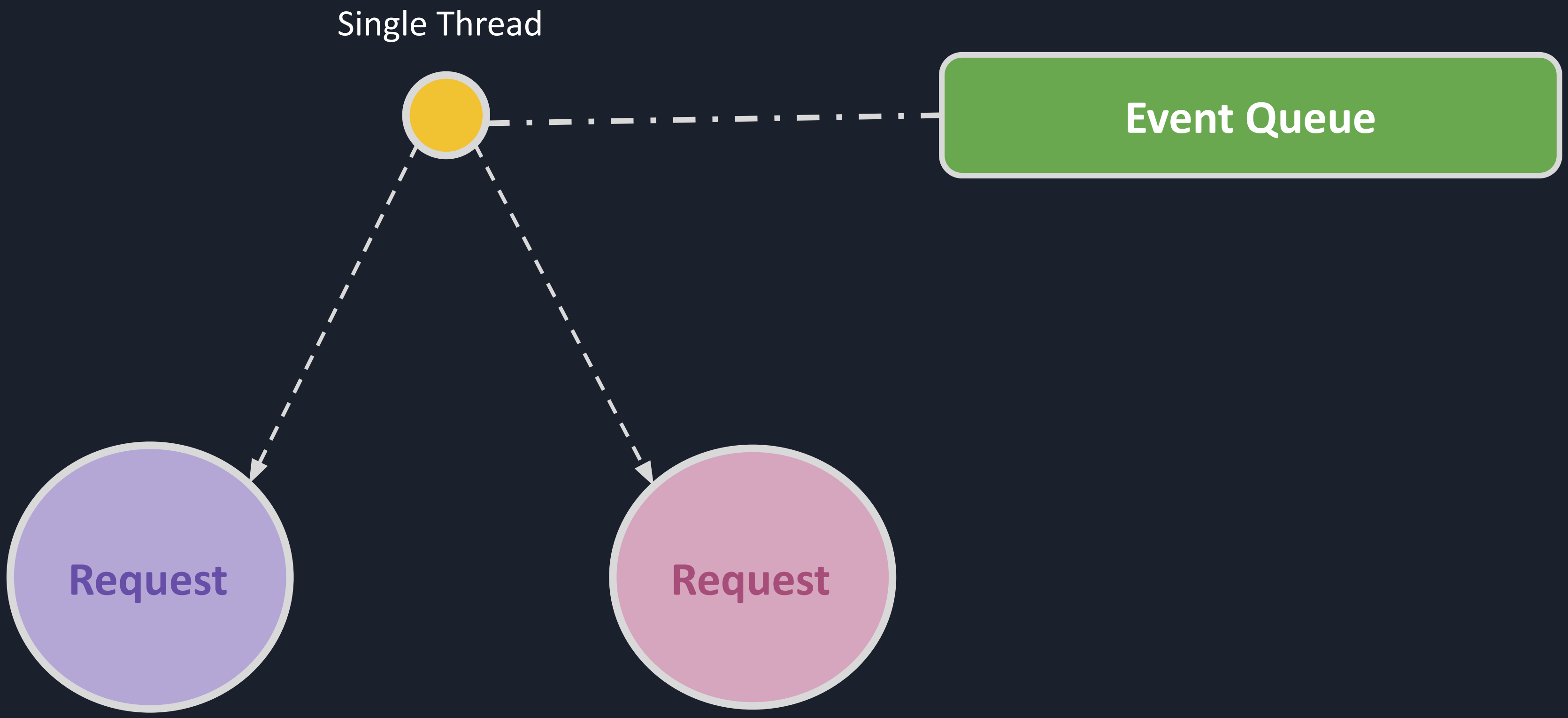
SERVER



All threads are busy!!!




SERVER
SERVER

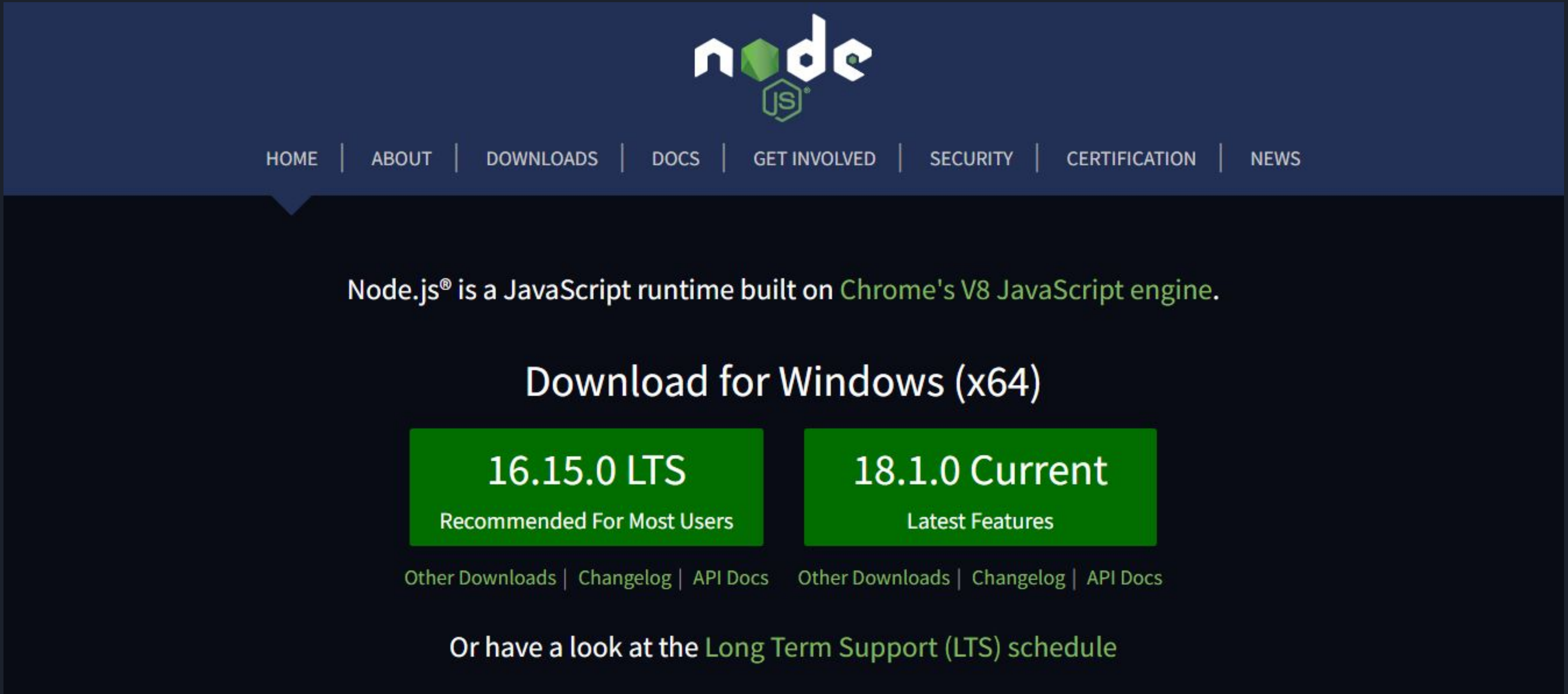

Single Thread
Event Queue
Request
Request

Drawback of Node.js

Node should not be used for CPU-intensive applications like video-encoding or image manipulation service because in such applications there are a lot of calculations that need to be done by the CPU and a very few operations that touch file system or the network.

Thus, node is ideal for building data intensive and real time applications.

Installing Node


HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | CERTIFICATION | NEWS

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

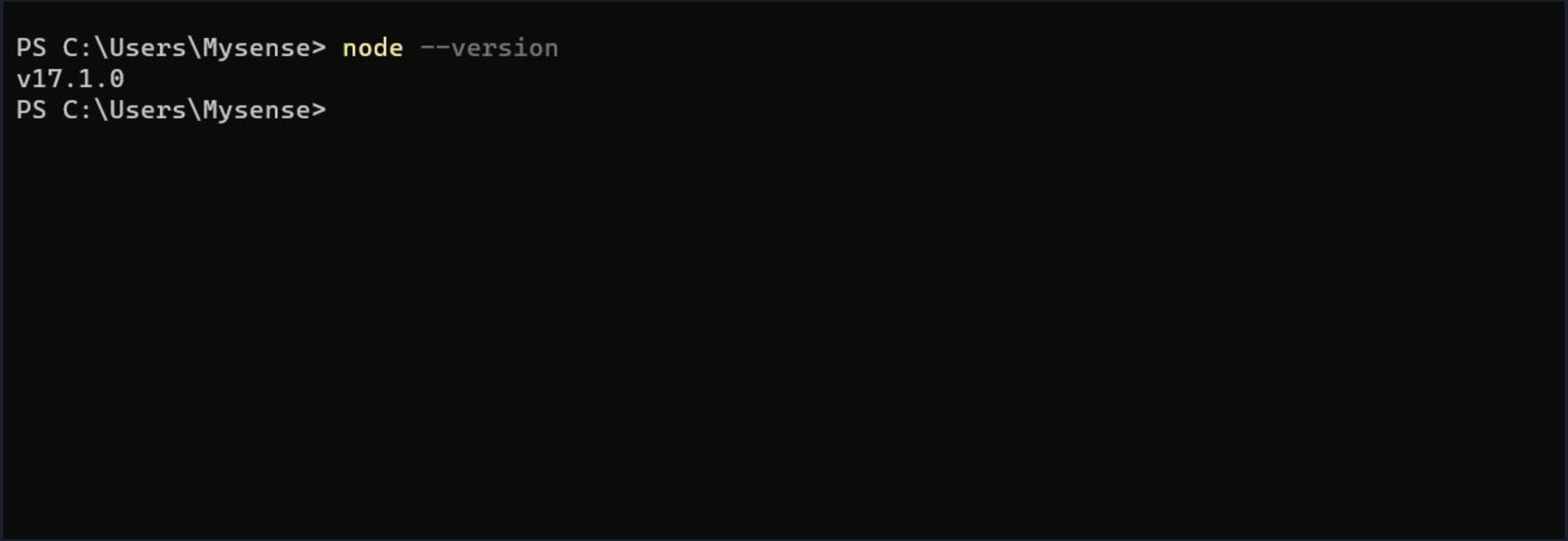
Download for Windows (x64)

16.15.0 LTS
Recommended For Most Users
18.1.0 Current
Latest Features
Other Downloads | Changelog | API Docs
Other Downloads | Changelog | API Docs

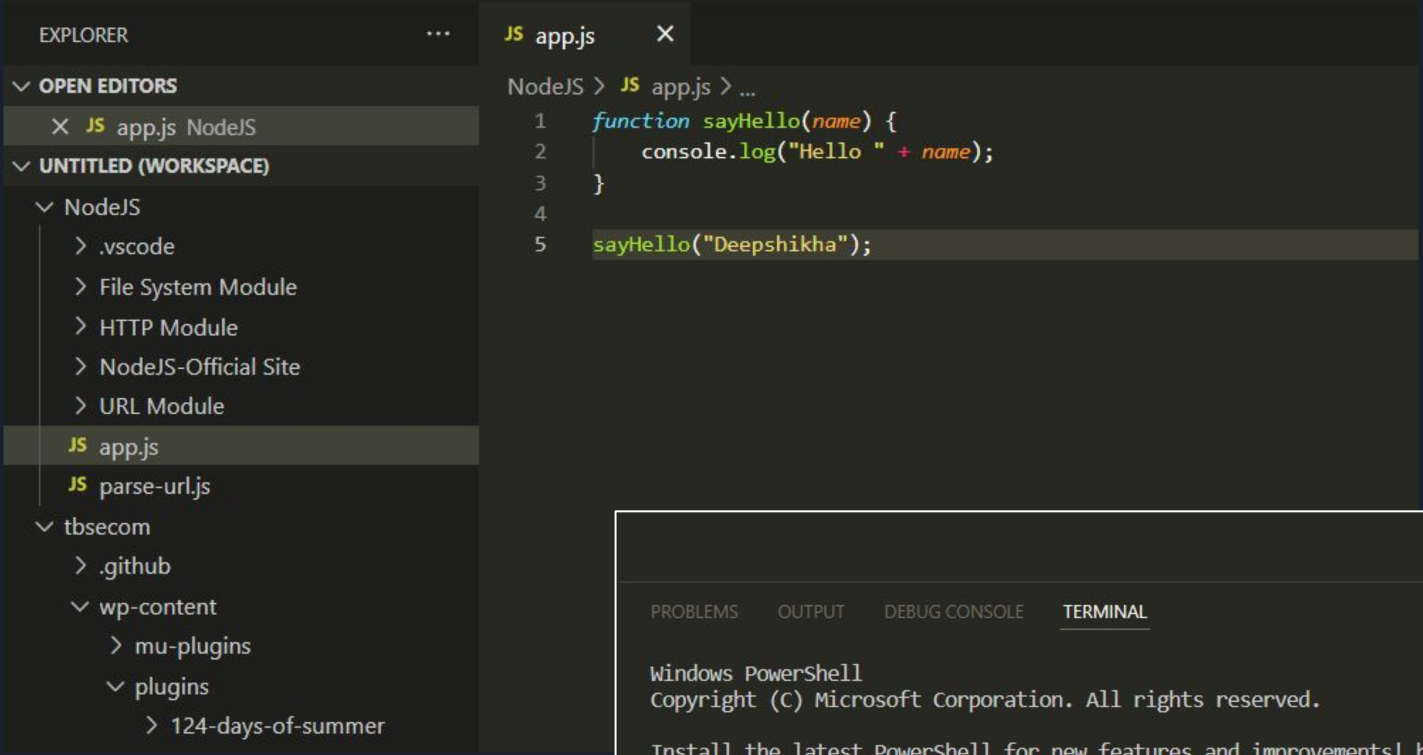
Or have a look at the Long Term Support (LTS) schedule

URL - https://nodejs.org/en/

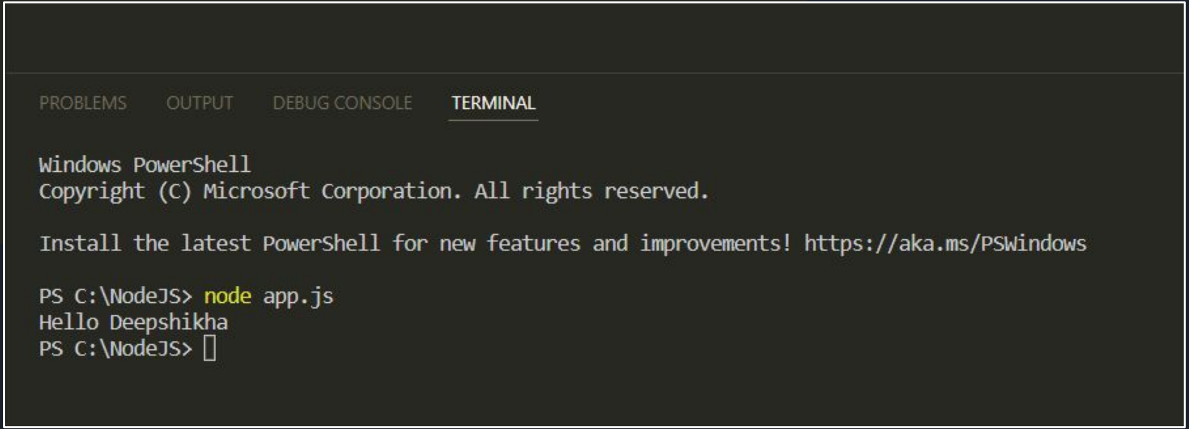
Terminal



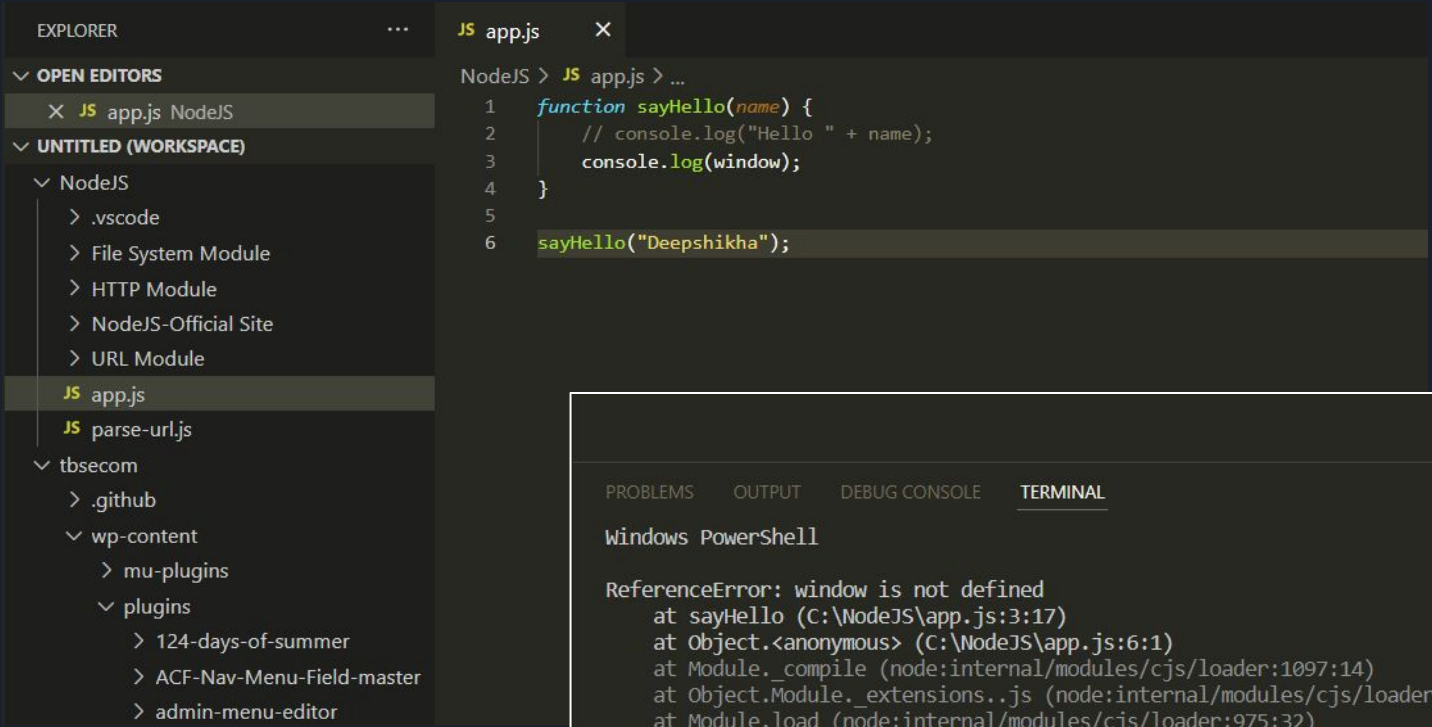
```
PS C:\Users\Mysense> node --version
v17.1.0
PS C:\Users\Mysense>
```



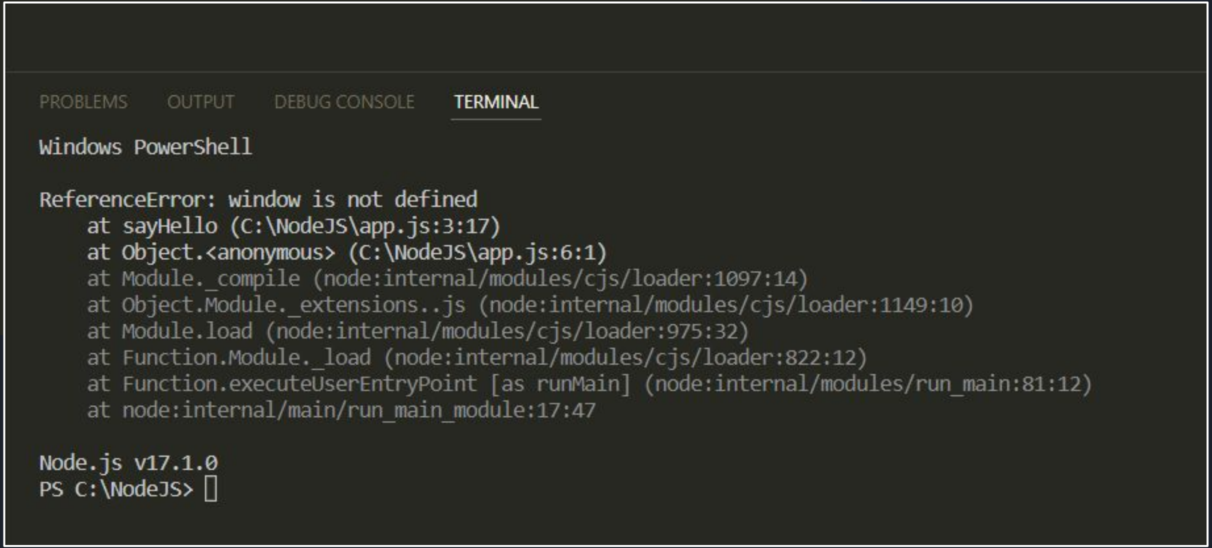
```javascript
function sayHello(name) {
    console.log("Hello " + name);
}

sayHello("Deepshikha");
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\NodeJS> node app.js
Hello Deepshikha
PS C:\NodeJS> 
```

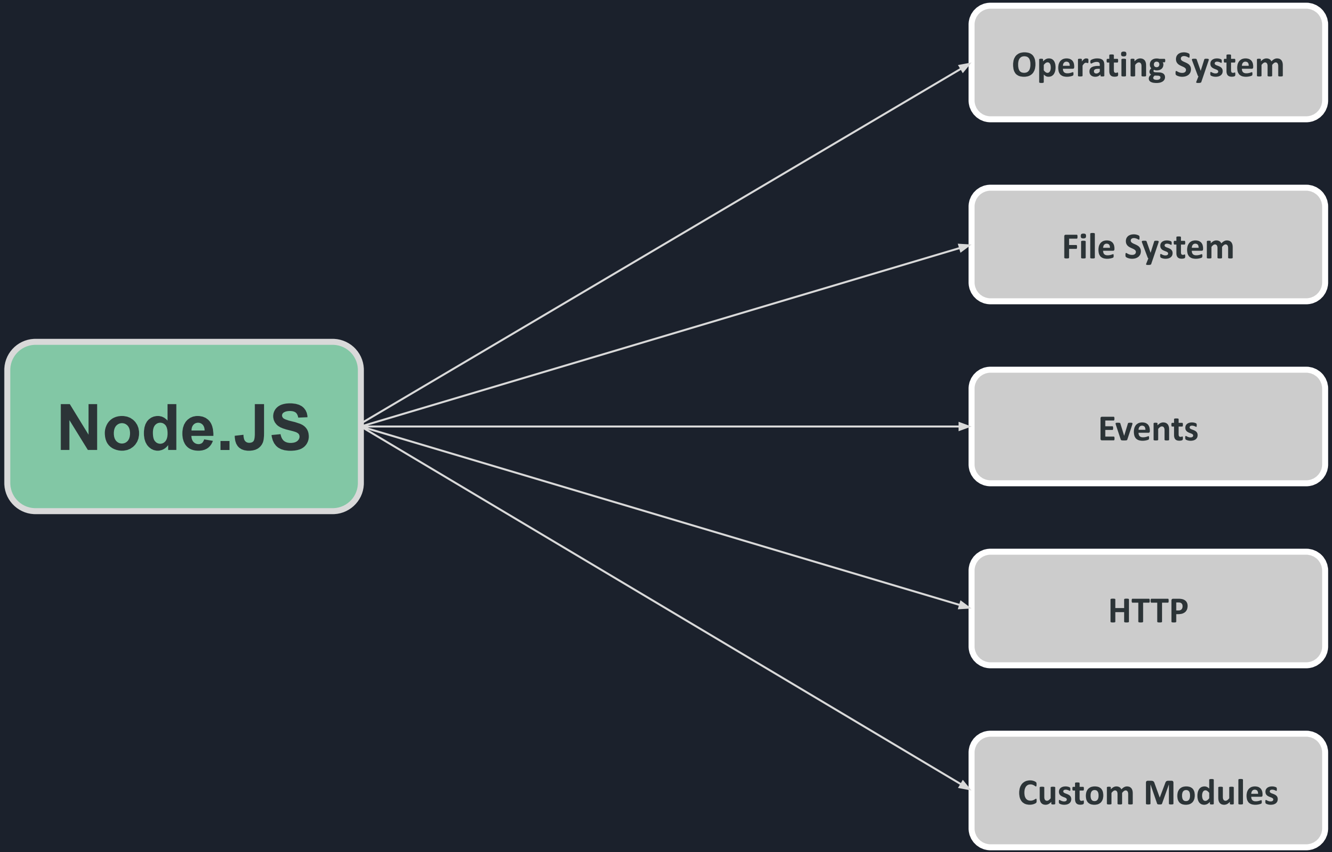


```
function sayHello(name) {
    // console.log("Hello " + name);
    console.log(window);
}

sayHello("Deepshikha");
```



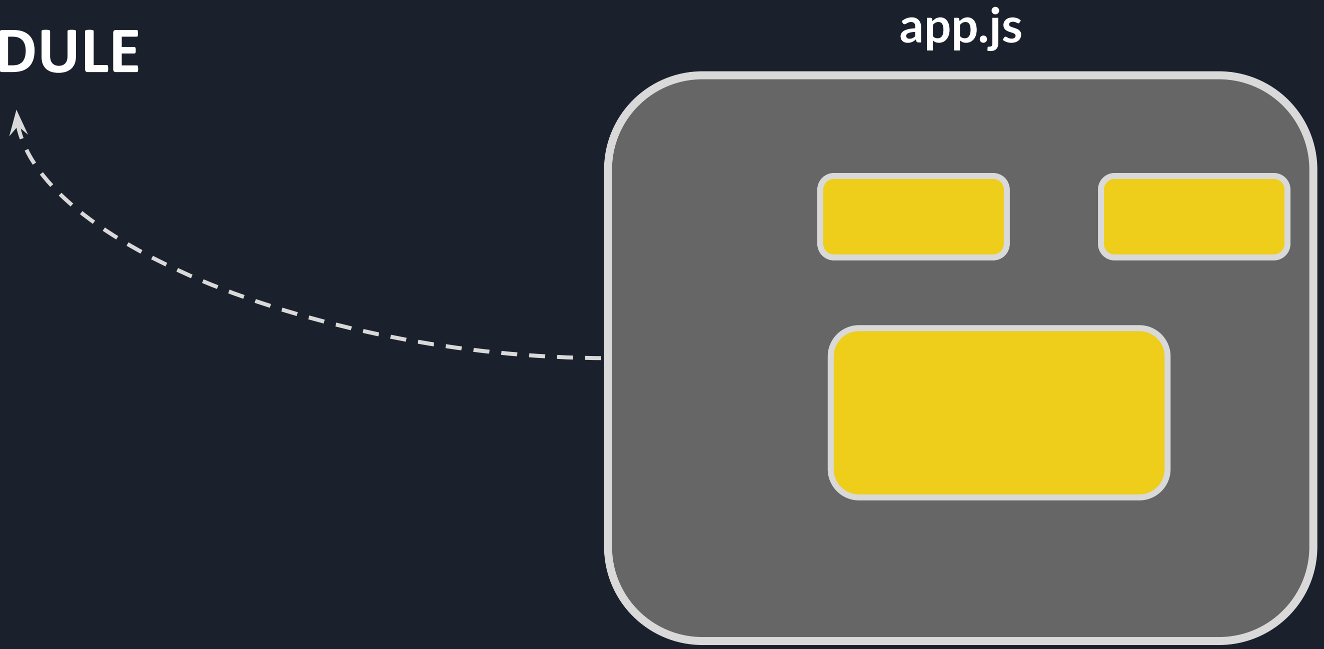
```
Windows PowerShell

ReferenceError: window is not defined
    at sayHello (C:\NodeJS\app.js:3:17)
    at Object.<anonymous> (C:\NodeJS\app.js:6:1)
    at Module._compile (node:internal/modules/cjs/loader:1097:14)
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1149:10)
    at Module.load (node:internal/modules/cjs/loader:975:32)
    at Function.Module._load (node:internal/modules/cjs/loader:822:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)
    at node:internal/main/run_main_module:17:47

Node.js v17.1.0
PS C:\NodeJS>
```

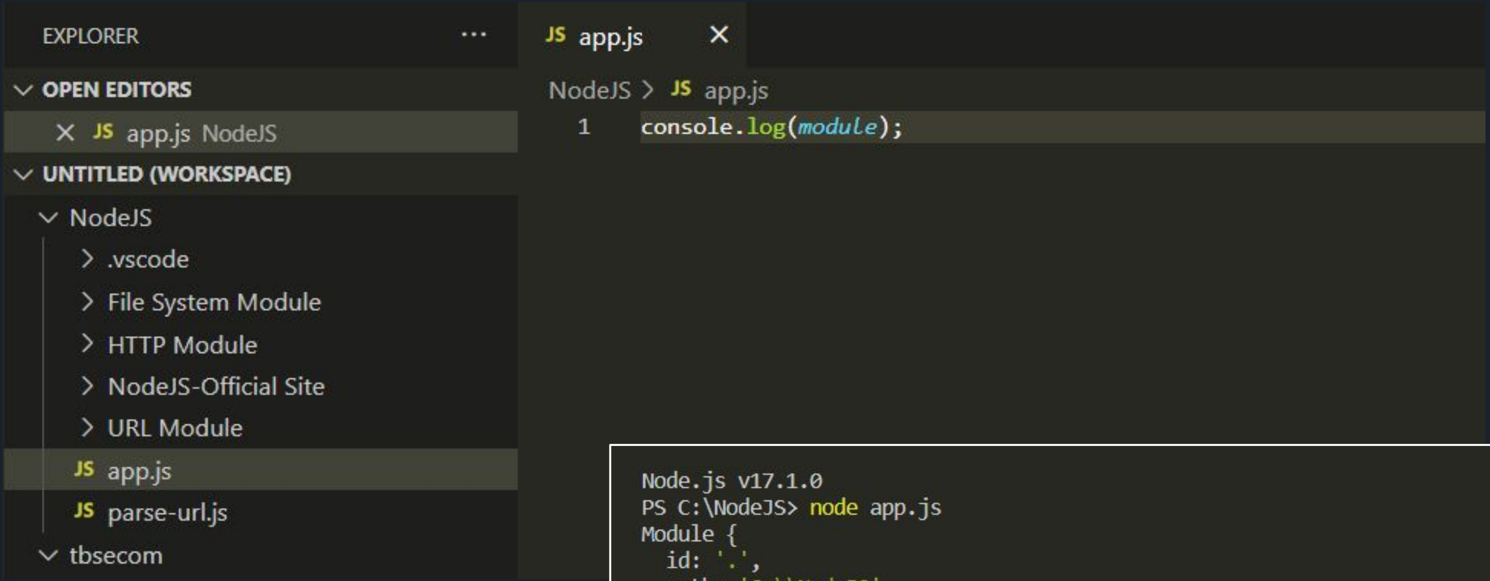
Node Module System


Node.JS
Operating System
File System
Events
HTTP
Custom Modules

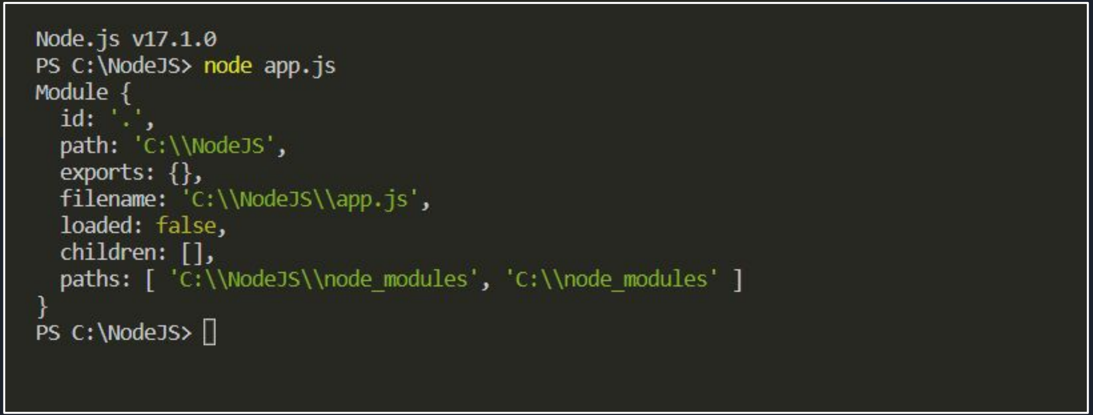
Module 1
Module 2
Module 3

MODULE



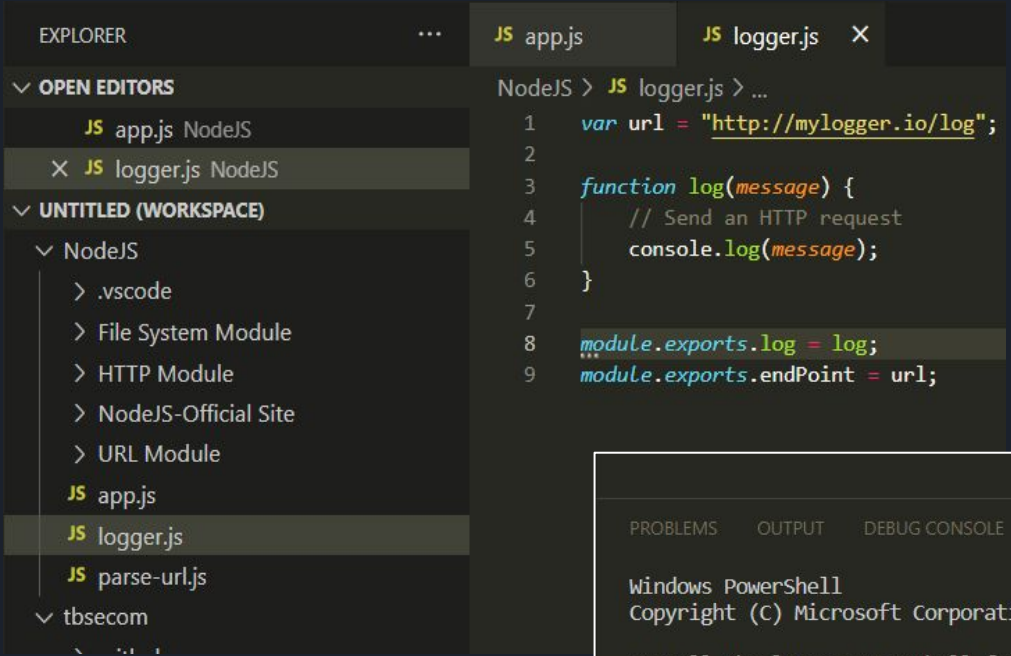


```javascript
console.log(module);
```

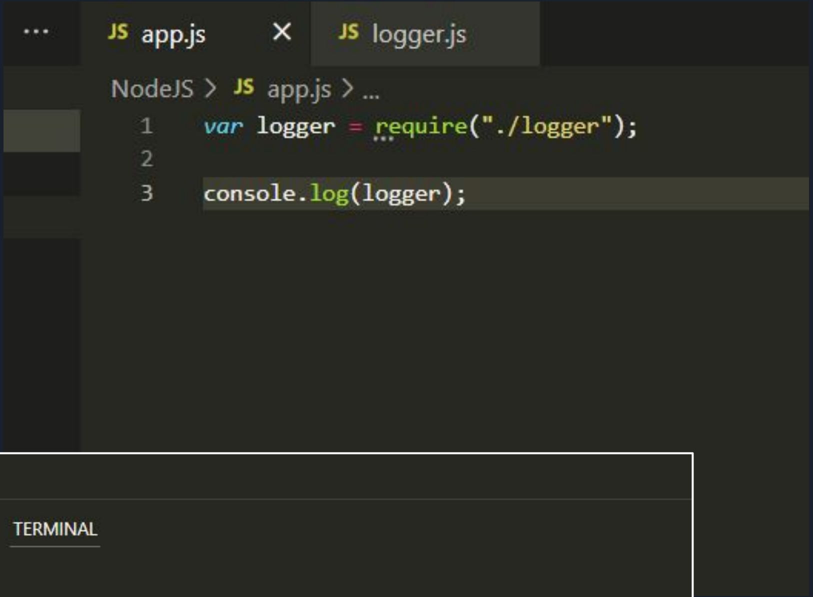


```
Node.js v17.1.0
PS C:\NodeJS> node app.js
Module {
  id: '.',
  path: 'C:\\NodeJS',
  exports: {},
  filename: 'C:\\NodeJS\\app.js',
  loaded: false,
  children: [],
  paths: [ 'C:\\NodeJS\\node_modules', 'C:\\node_modules' ]
}
PS C:\NodeJS> 
```

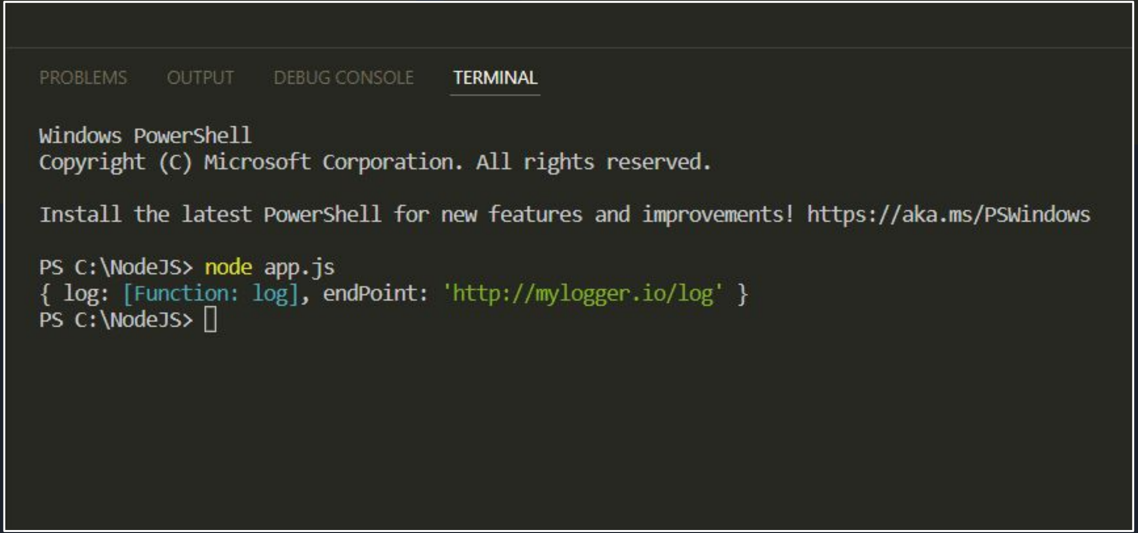
Creating a Module



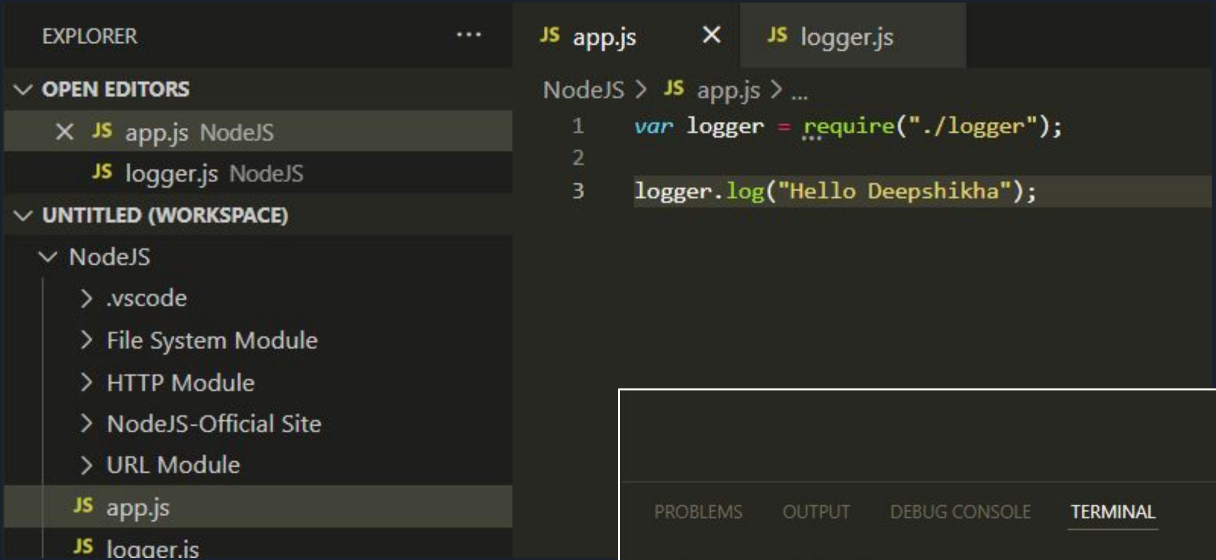
```javascript
var url = "http://mylogger.io/log";

function log(message) {
    // Send an HTTP request
    console.log(message);
}

module.exports.log = log;
module.exports.endPoint = url;
```



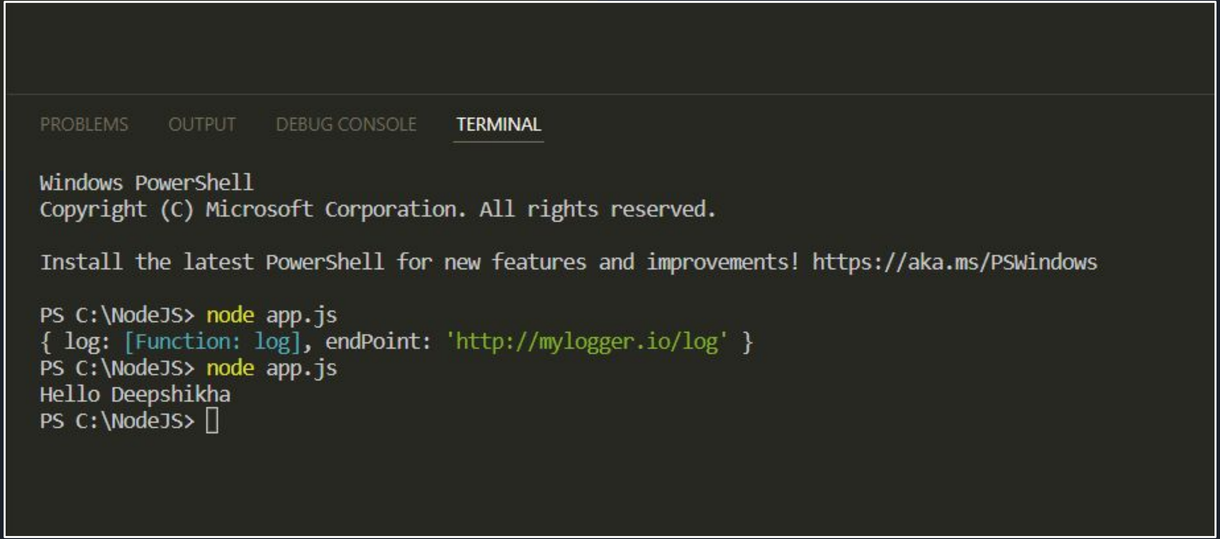
```javascript
var logger = require("./logger");

console.log(logger);
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\NodeJS> node app.js
{ log: [Function: log], endPoint: 'http://mylogger.io/log' }
PS C:\NodeJS> 
```

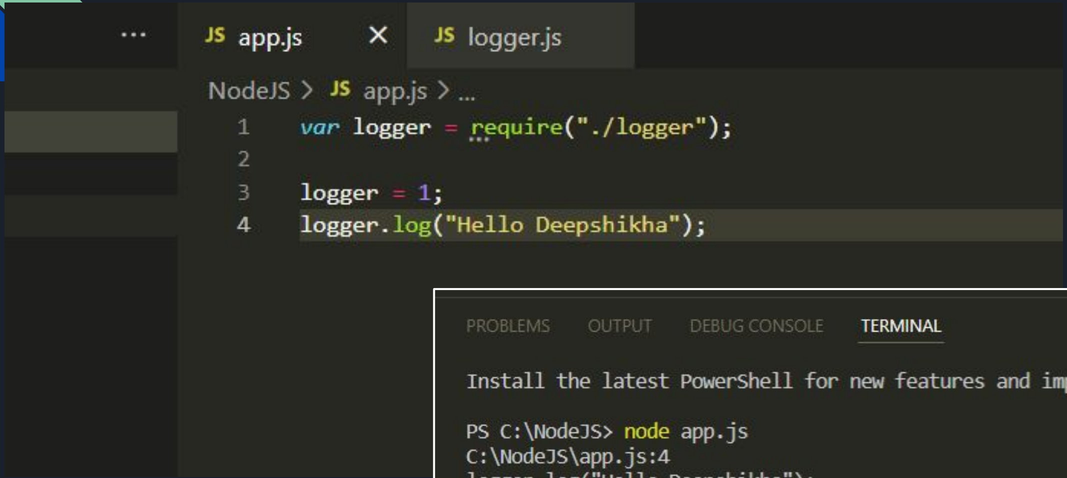


```javascript
var logger = require("./logger");

logger.log("Hello Deepshikha");
```

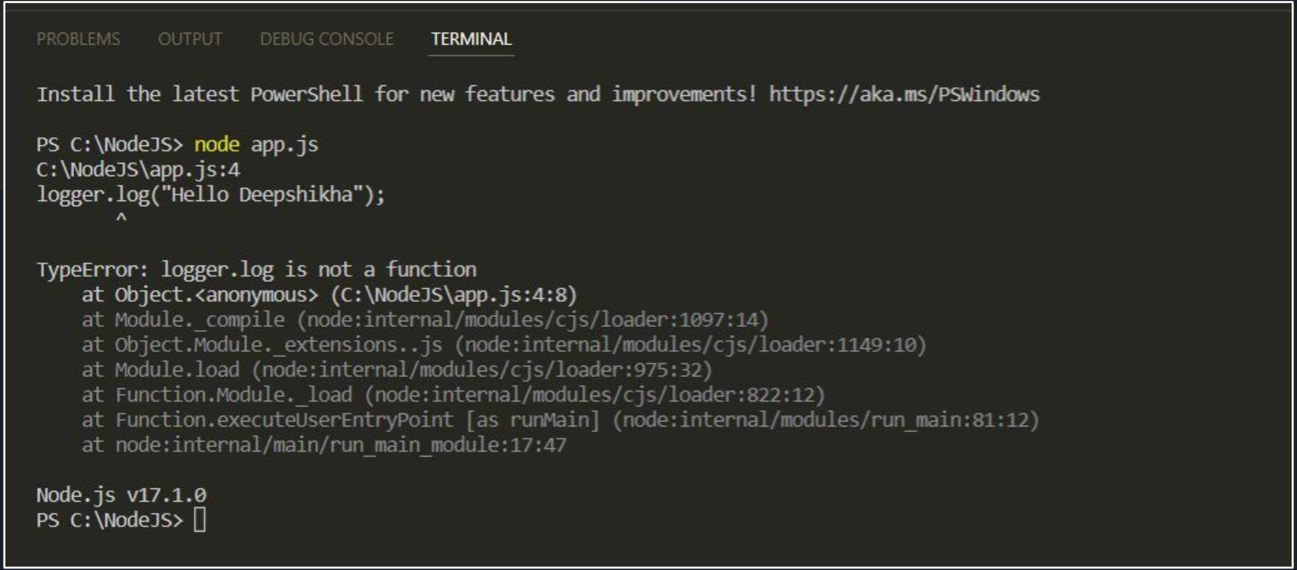


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\NodeJS> node app.js
{ log: [Function: log], endPoint: 'http://mylogger.io/log' }
PS C:\NodeJS> node app.js
Hello Deepshikha
PS C:\NodeJS> 
```

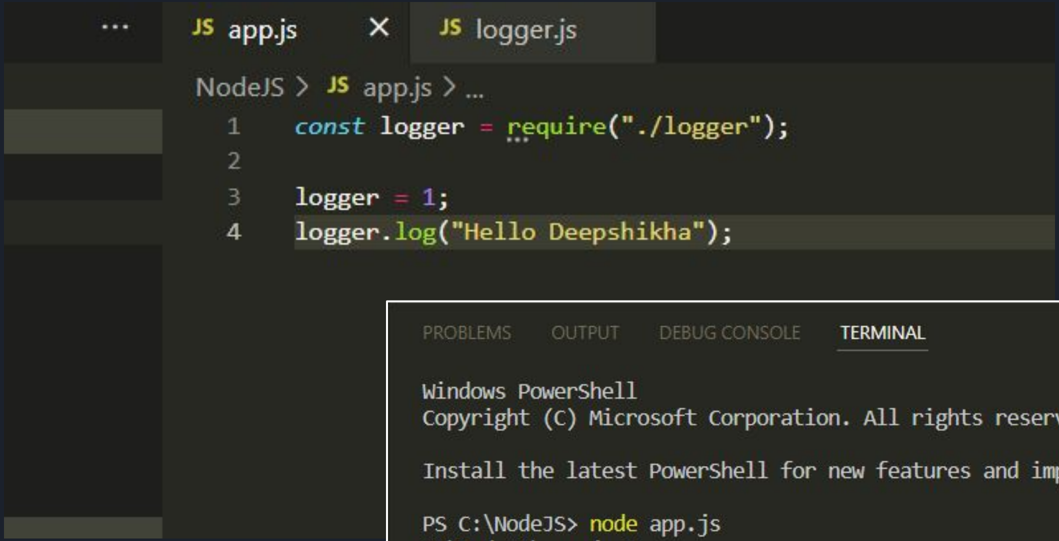
Using **const** instead of **var**



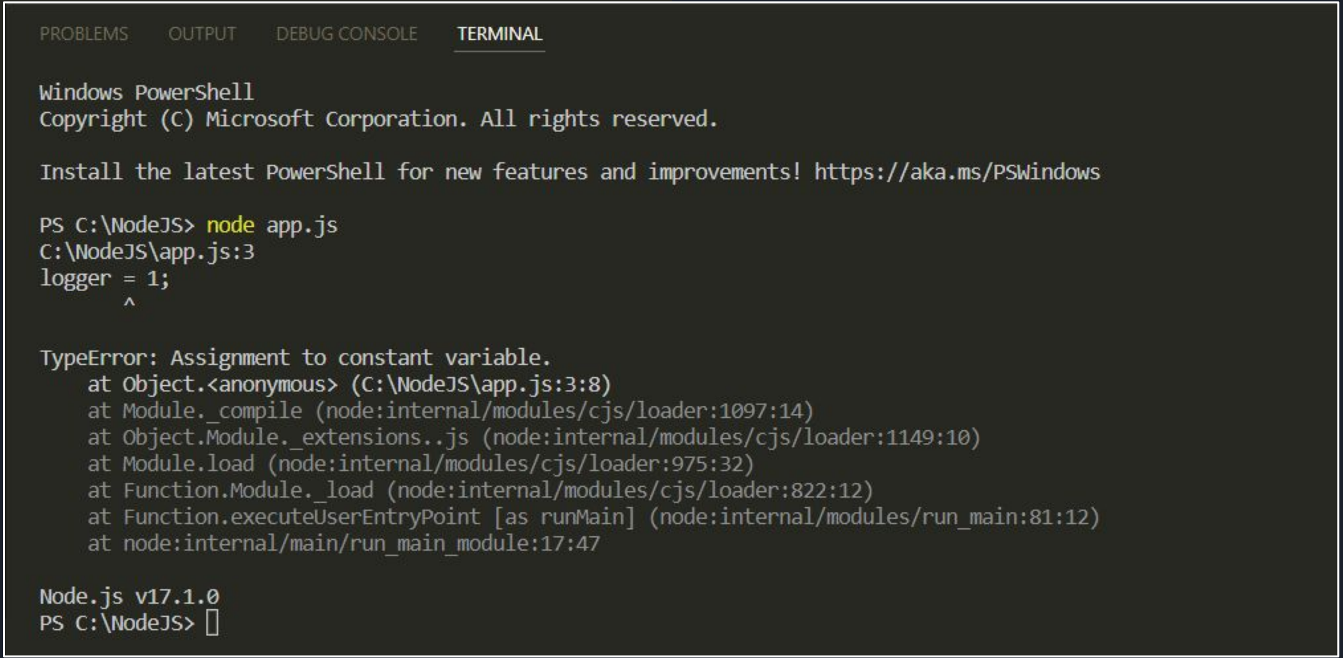
```
var logger = require("./logger");

logger = 1;
logger.log("Hello Deepshikha");
```



```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\NodeJS> node app.js
C:\NodeJS\app.js:4
logger.log("Hello Deepshikha");
       ^

TypeError: logger.log is not a function
    at Object.<anonymous> (C:\NodeJS\app.js:4:8)
    at Module._compile (node:internal/modules/cjs/loader:1097:14)
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1149:10)
    at Module.load (node:internal/modules/cjs/loader:975:32)
    at Function.Module._load (node:internal/modules/cjs/loader:822:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)
    at node:internal/main/run_main_module:17:47

Node.js v17.1.0
PS C:\NodeJS> 
```

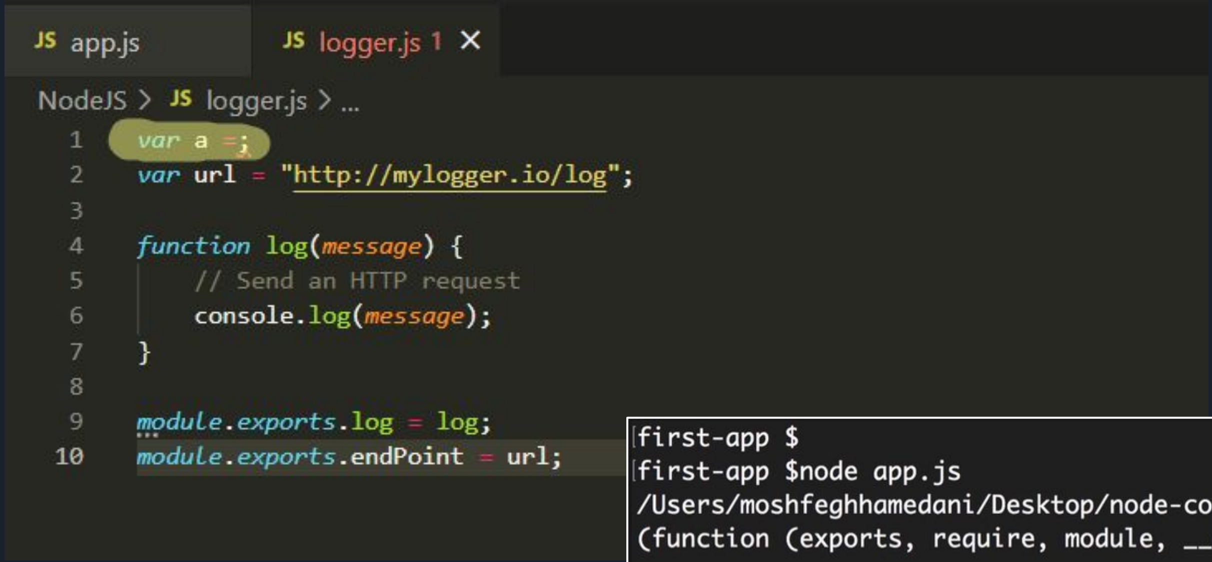


```
const logger = require("./logger");

logger = 1;
logger.log("Hello Deepshikha");
```

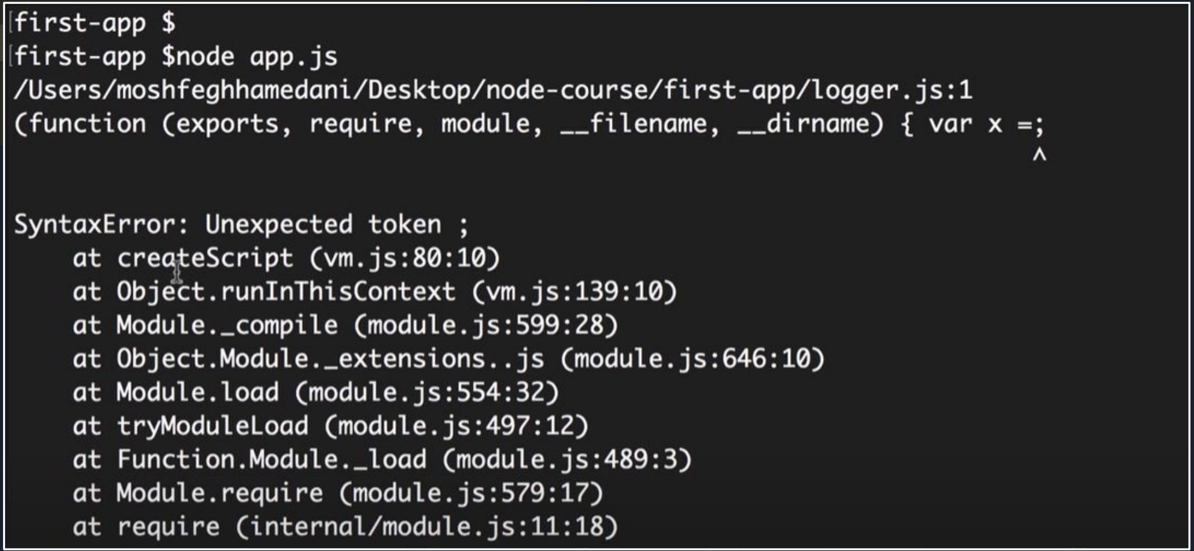


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\NodeJS> node app.js
C:\NodeJS\app.js:3
logger = 1;
       ^

TypeError: Assignment to constant variable.
    at Object.<anonymous> (C:\NodeJS\app.js:3:8)
    at Module._compile (node:internal/modules/cjs/loader:1097:14)
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1149:10)
    at Module.load (node:internal/modules/cjs/loader:975:32)
    at Function.Module._load (node:internal/modules/cjs/loader:822:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)
    at node:internal/main/run_main_module:17:47

Node.js v17.1.0
PS C:\NodeJS> 
```

Module Wrapper Function

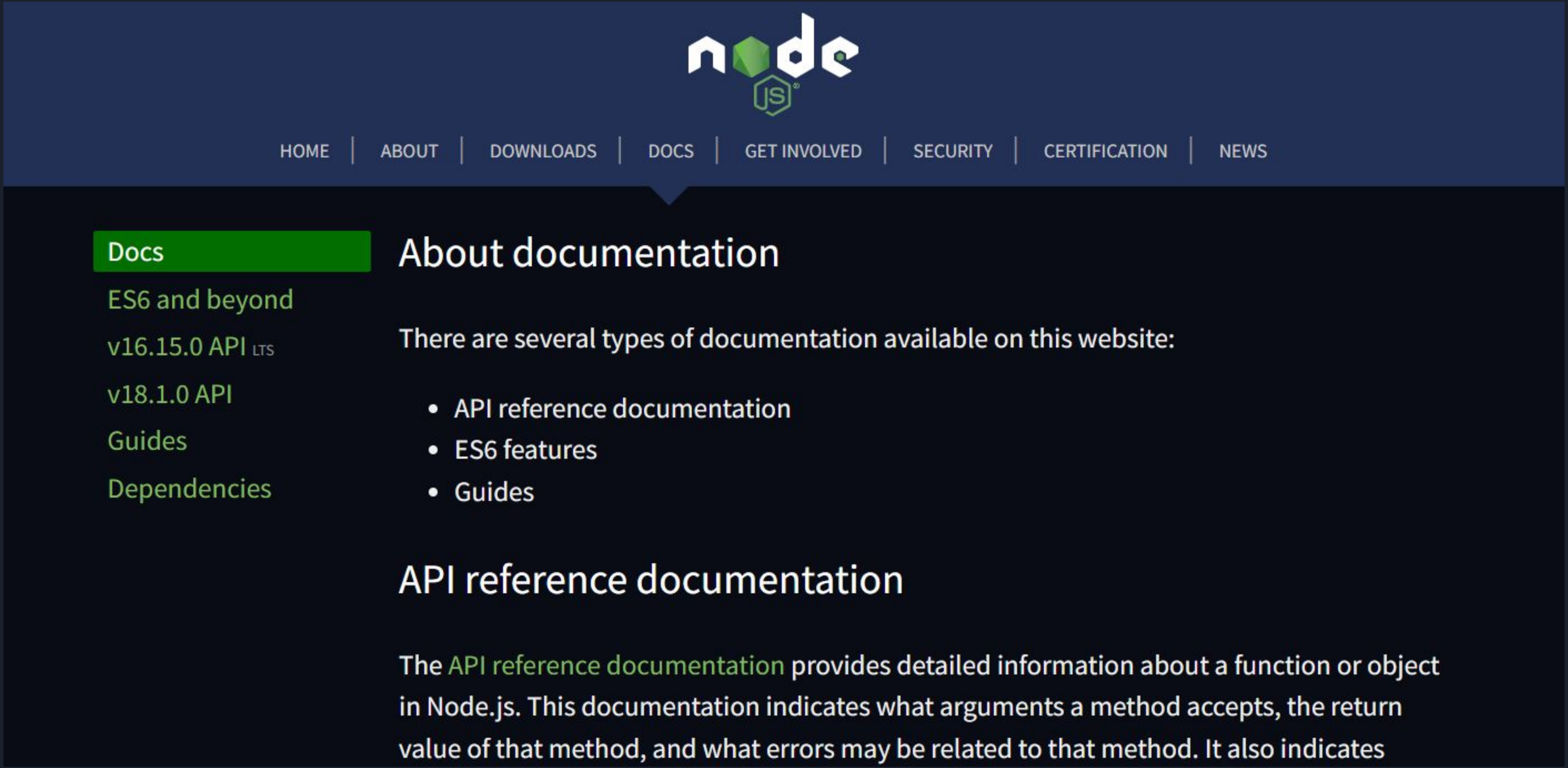


```javascript
var a =;
var url = "http://mylogger.io/log";

function log(message) {
    // Send an HTTP request
    console.log(message);
}

module.exports.log = log;
module.exports.endPoint = url;
```



```
first-app $
first-app $node app.js
/Users/moshfeghhamedani/Desktop/node-course/first-app/logger.js:1
(function (exports, require, module, __filename, __dirname) { var x =;
                                                                       ^


SyntaxError: Unexpected token ;
    at createScript (vm.js:80:10)
    at Object.runInThisContext (vm.js:139:10)
    at Module._compile (module.js:599:28)
    at Object.Module._extensions..js (module.js:646:10)
    at Module.load (module.js:554:32)
    at tryModuleLoad (module.js:497:12)
    at Function.Module._load (module.js:489:3)
    at Module.require (module.js:579:17)
    at require (internal/module.js:11:18)
```

In-Built Modules



About documentation

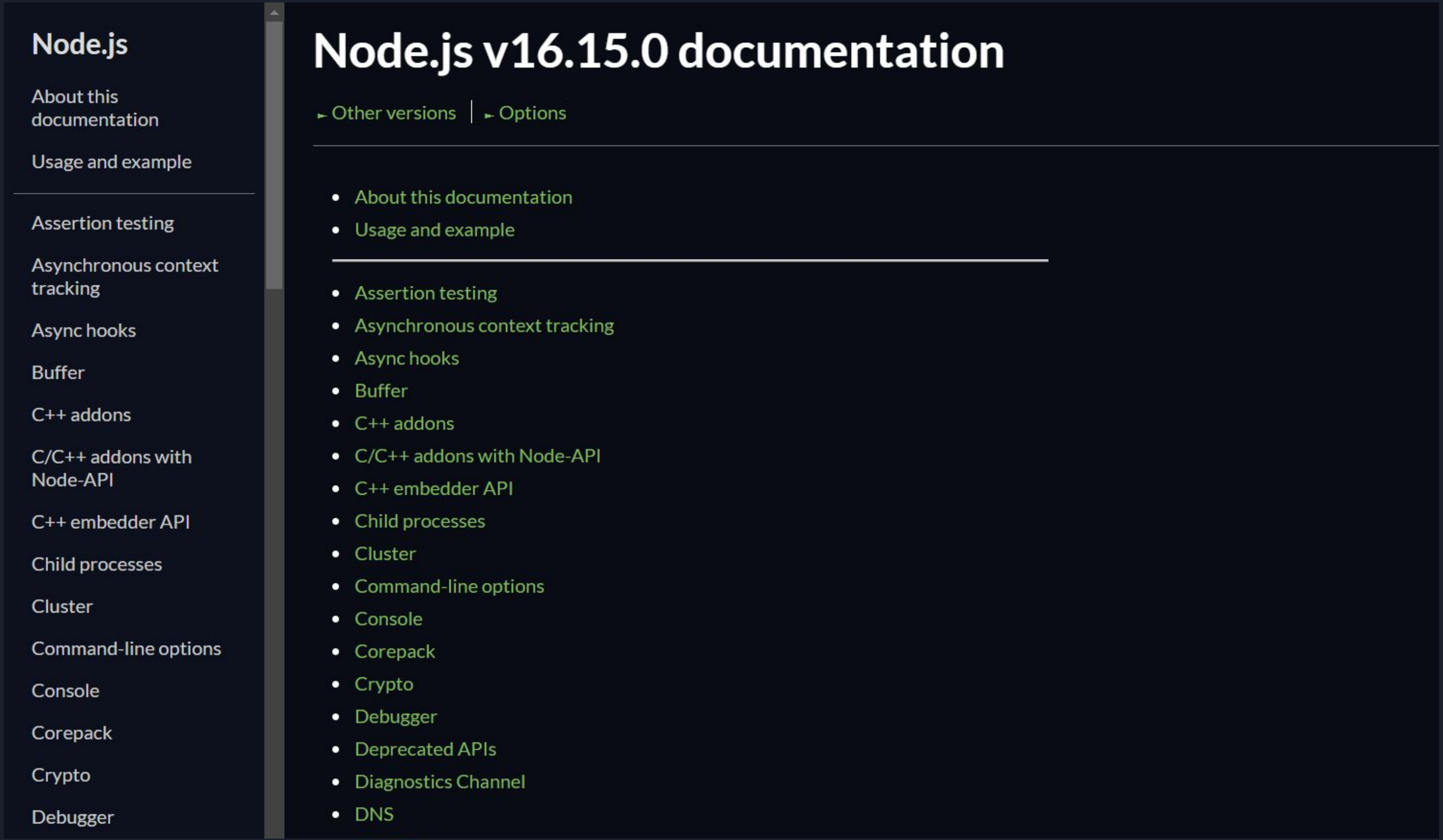
There are several types of documentation available on this website:

- API reference documentation
- ES6 features
- Guides

API reference documentation

The API reference documentation provides detailed information about a function or object in Node.js. This documentation indicates what arguments a method accepts, the return value of that method, and what errors may be related to that method. It also indicates

URL - https://nodejs.org/en/docs/



Node.js

About this documentation

Usage and example

Assertion testing

Asynchronous context tracking

Async hooks

Buffer

C++ addons

C/C++ addons with Node-API

C++ embedder API

Child processes

Cluster

Command-line options

Console

Corepack

Crypto

Debugger

Node.js v16.15.0 documentation

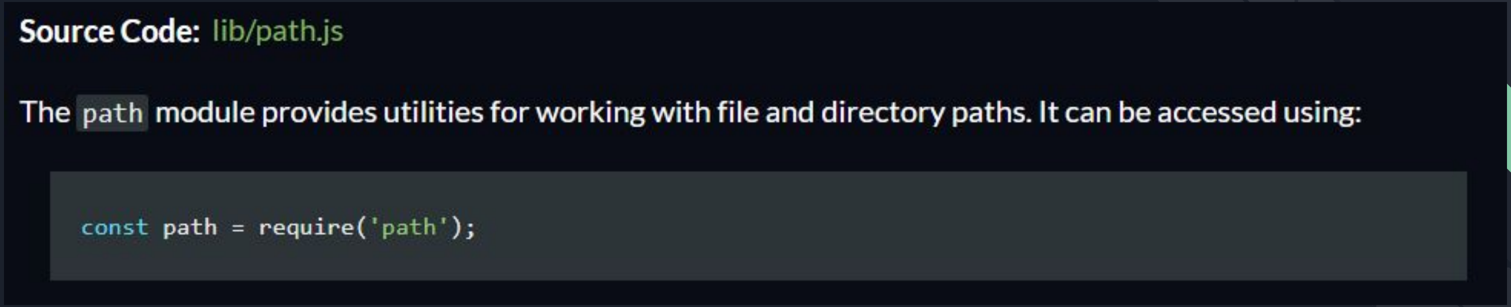
► Other versions | ► Options

- About this documentation
- Usage and example

- Assertion testing
- Asynchronous context tracking
- Async hooks
- Buffer
- C++ addons
- C/C++ addons with Node-API
- C++ embedder API
- Child processes
- Cluster
- Command-line options
- Console
- Corepack
- Crypto
- Debugger
- Deprecated APIs
- Diagnostics Channel
- DNS

URL - https://nodejs.org/dist/latest-v16.x/docs/api/

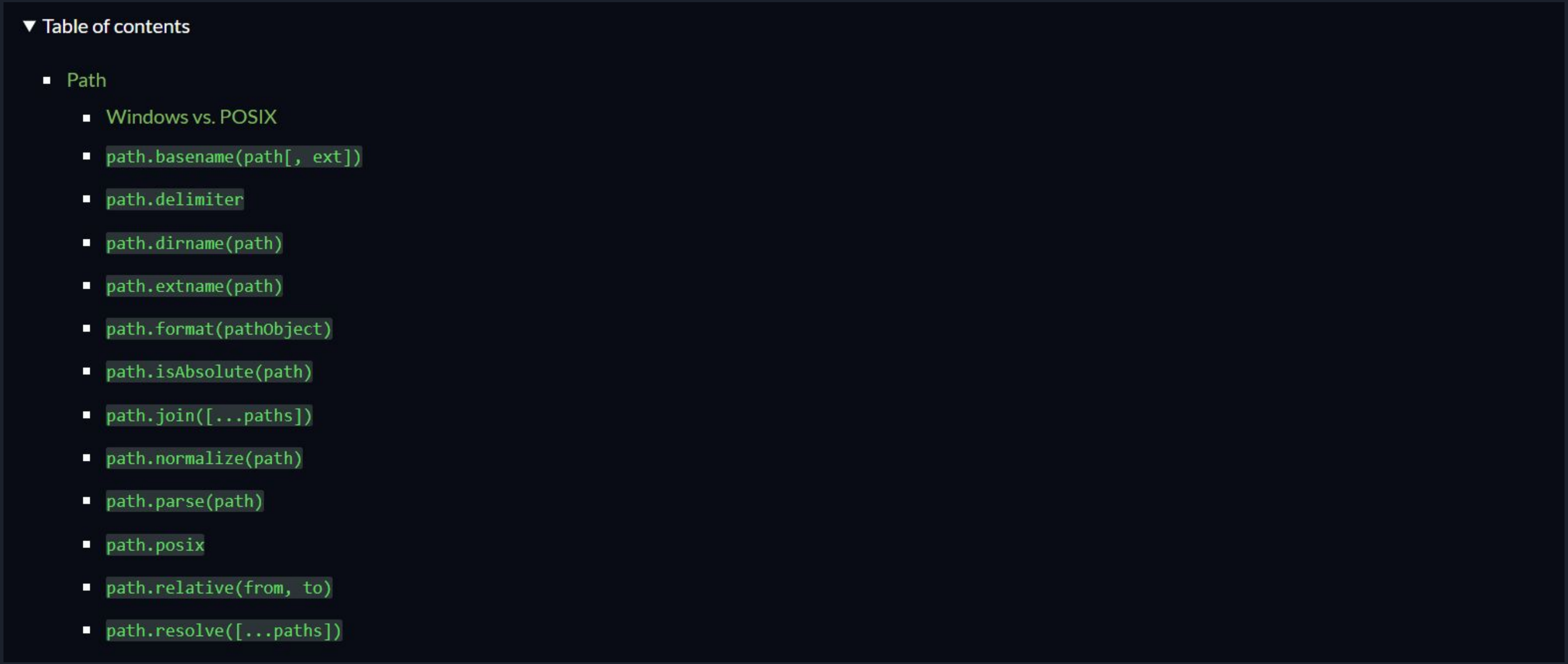
Path Module



Source Code: lib/path.js

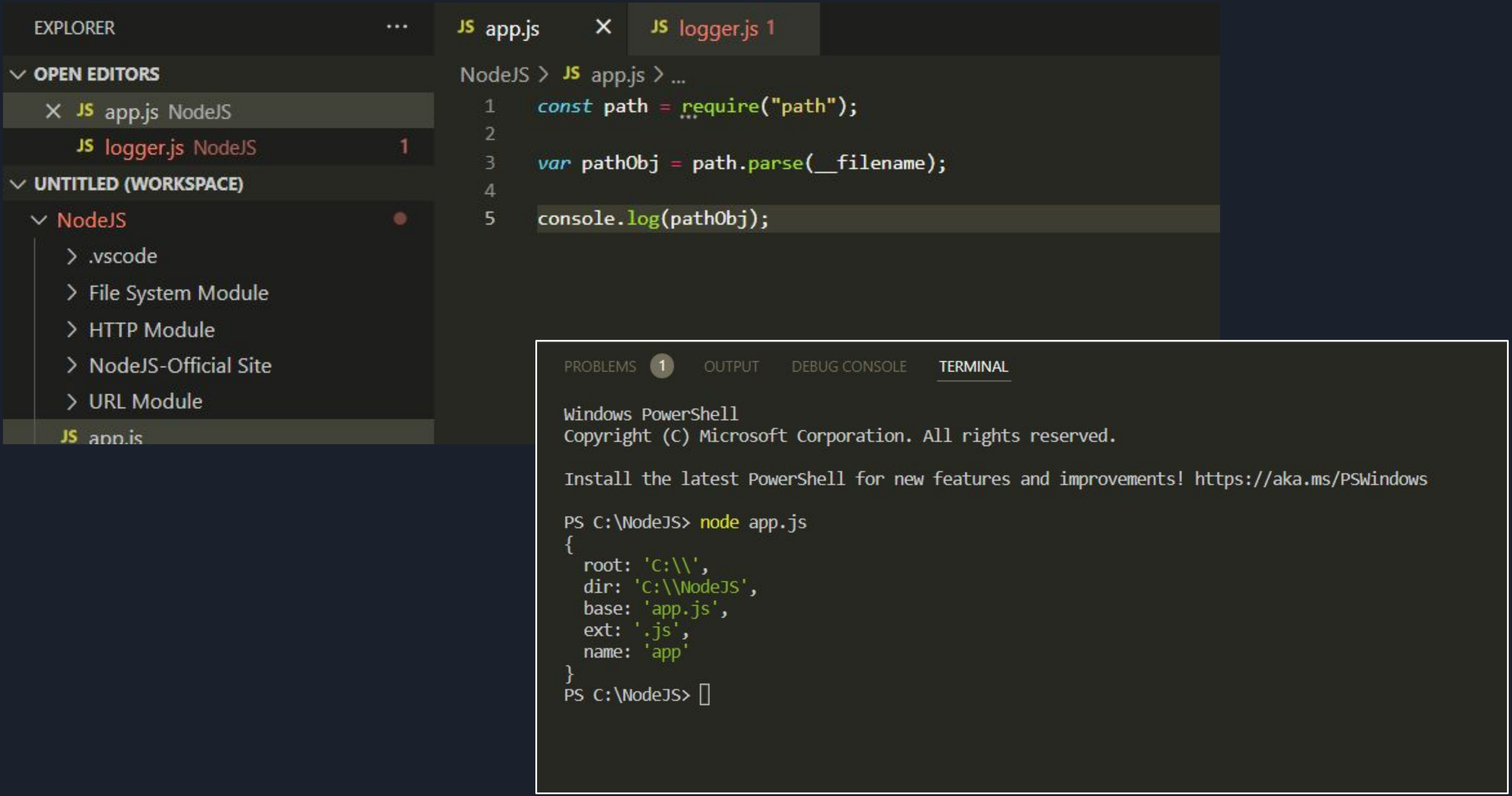
The `path` module provides utilities for working with file and directory paths. It can be accessed using:

```
const path = require('path');
```



▼ Table of contents

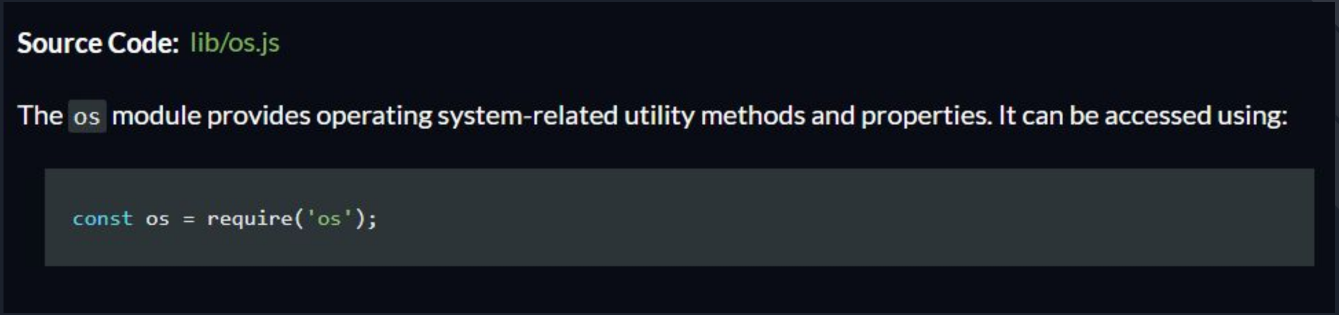
URL - https://nodejs.org/dist/latest-v16.x/docs/api/path.html



```
const path = require("path");

var pathObj = path.parse(__filename);

console.log(pathObj);
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\NodeJS> node app.js
{
  root: 'C:\\',
  dir: 'C:\\NodeJS',
  base: 'app.js',
  ext: '.js',
  name: 'app'
}
PS C:\NodeJS> 
```

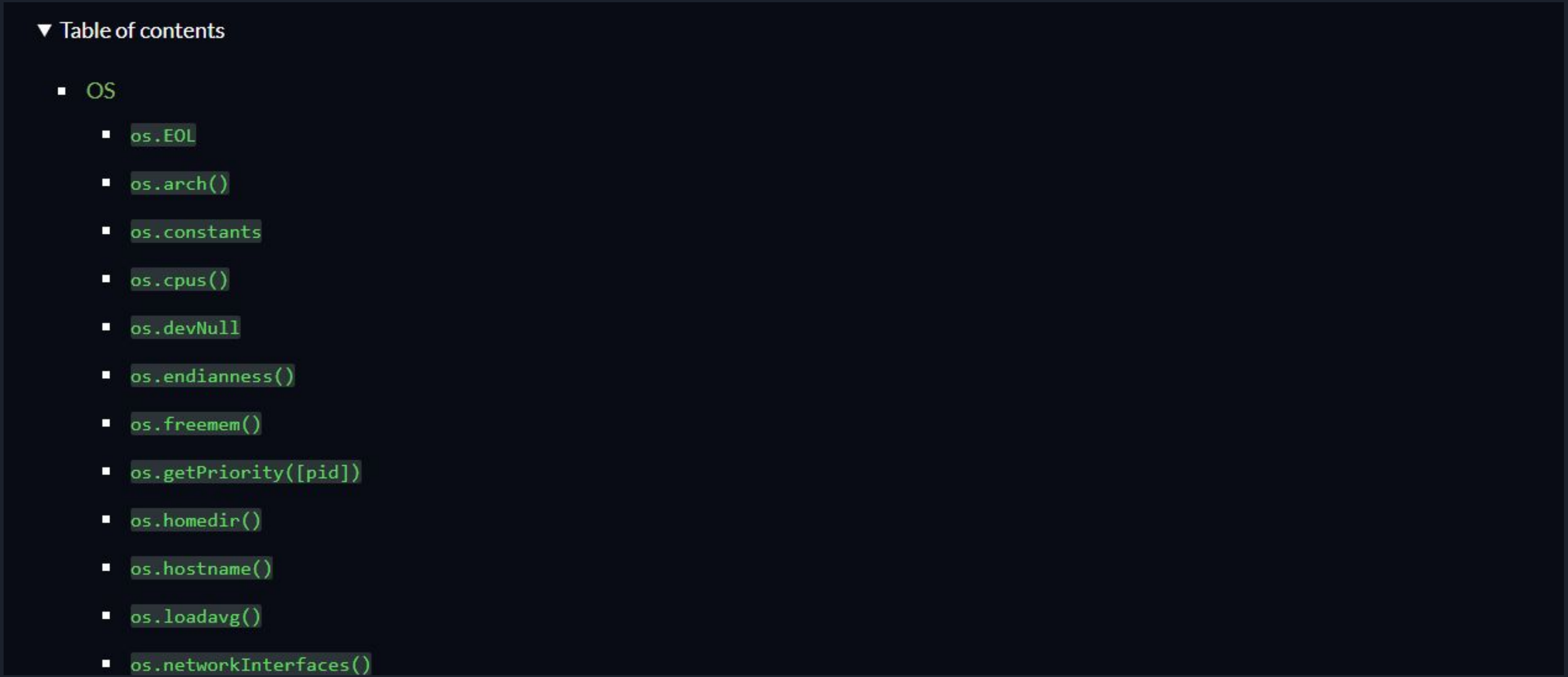
OS Module



Source Code: lib/os.js

The `os` module provides operating system-related utility methods and properties. It can be accessed using:

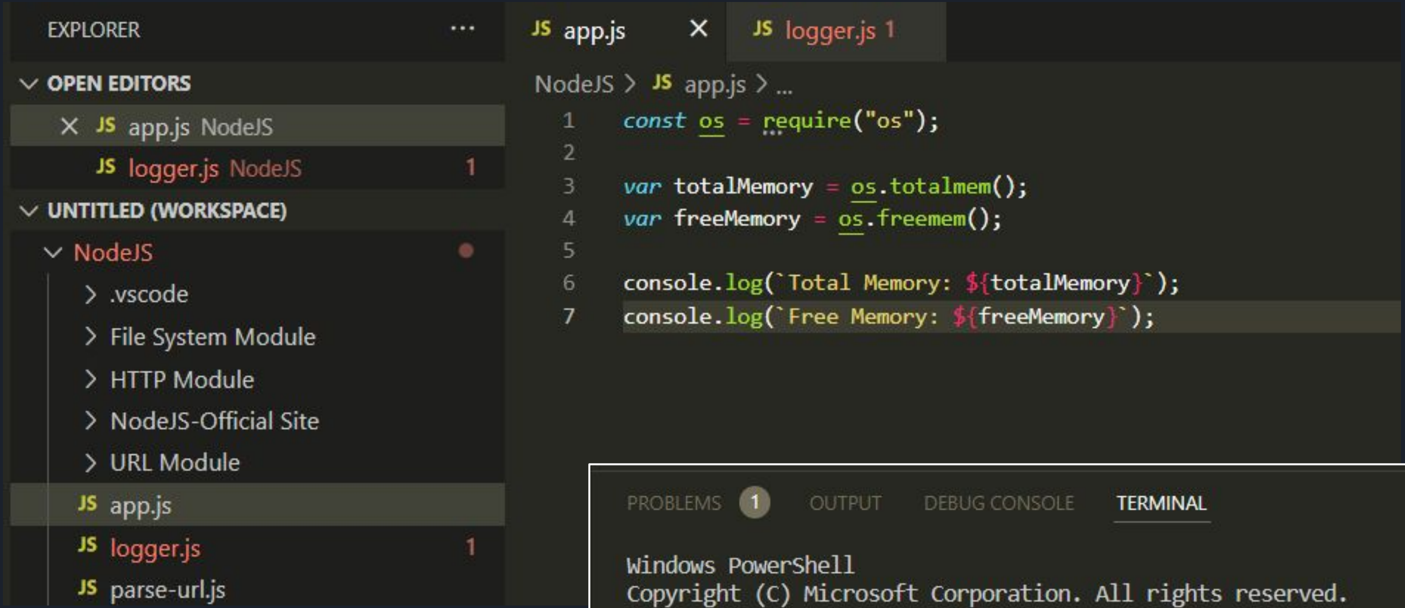
```
const os = require('os');
```



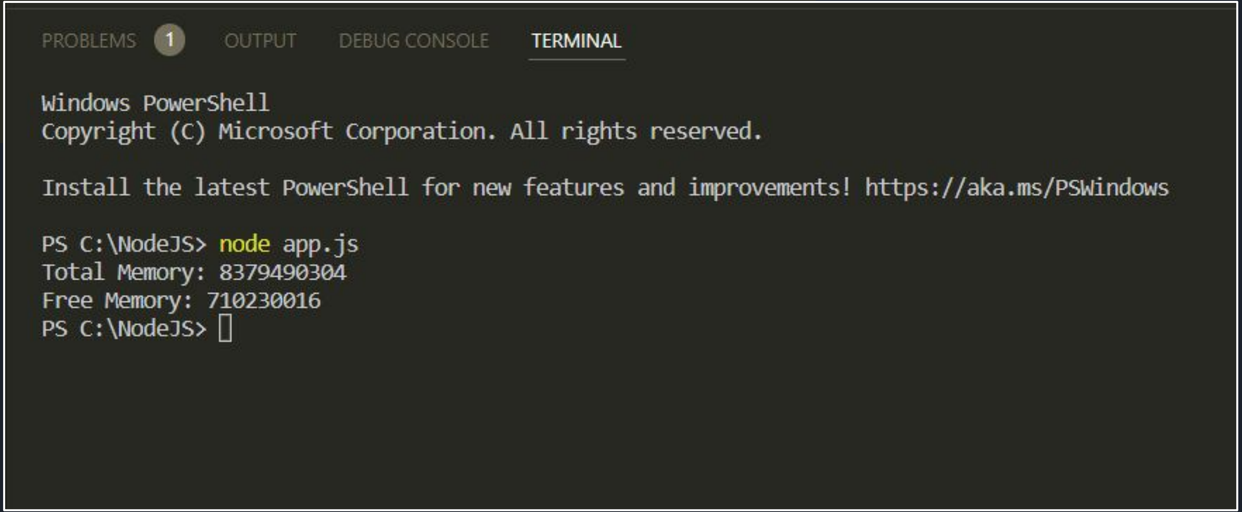
▼ Table of contents

- OS
  - `os.EOL`
  - `os.arch()`
  - `os.constants`
  - `os.cpus()`
  - `os.devNull`
  - `os.endianness()`
  - `os.freemem()`
  - `os.getPriority([pid])`
  - `os.homedir()`
  - `os.hostname()`
  - `os.loadavg()`
  - `os.networkInterfaces()`

URL - https://nodejs.org/dist/latest-v16.x/docs/api/os.html



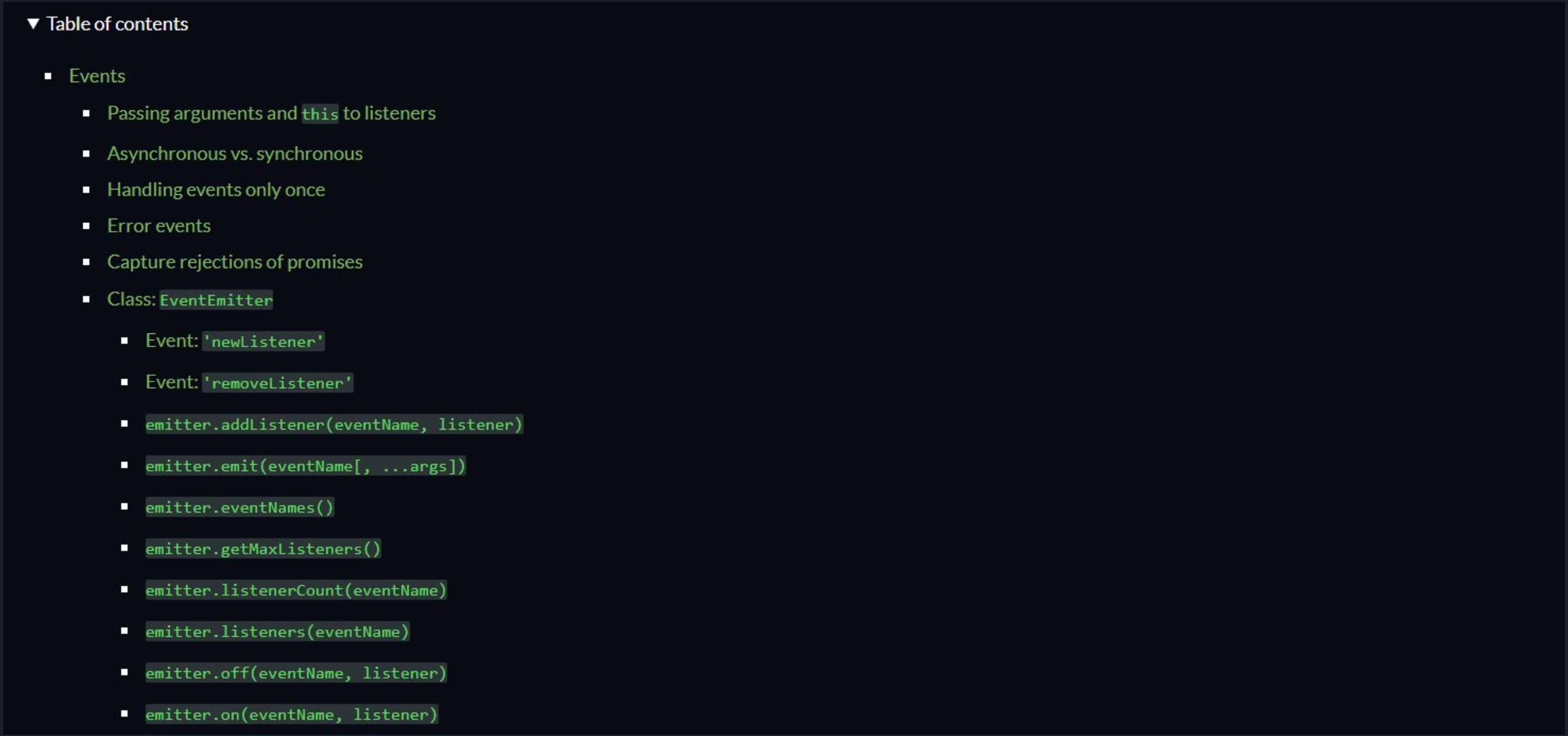
```javascript
const os = require("os");

var totalMemory = os.totalmem();
var freeMemory = os.freemem();

console.log(`Total Memory: ${totalMemory}`);
console.log(`Free Memory: ${freeMemory}`);
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\NodeJS> node app.js
Total Memory: 8379490304
Free Memory: 710230016
PS C:\NodeJS>
```

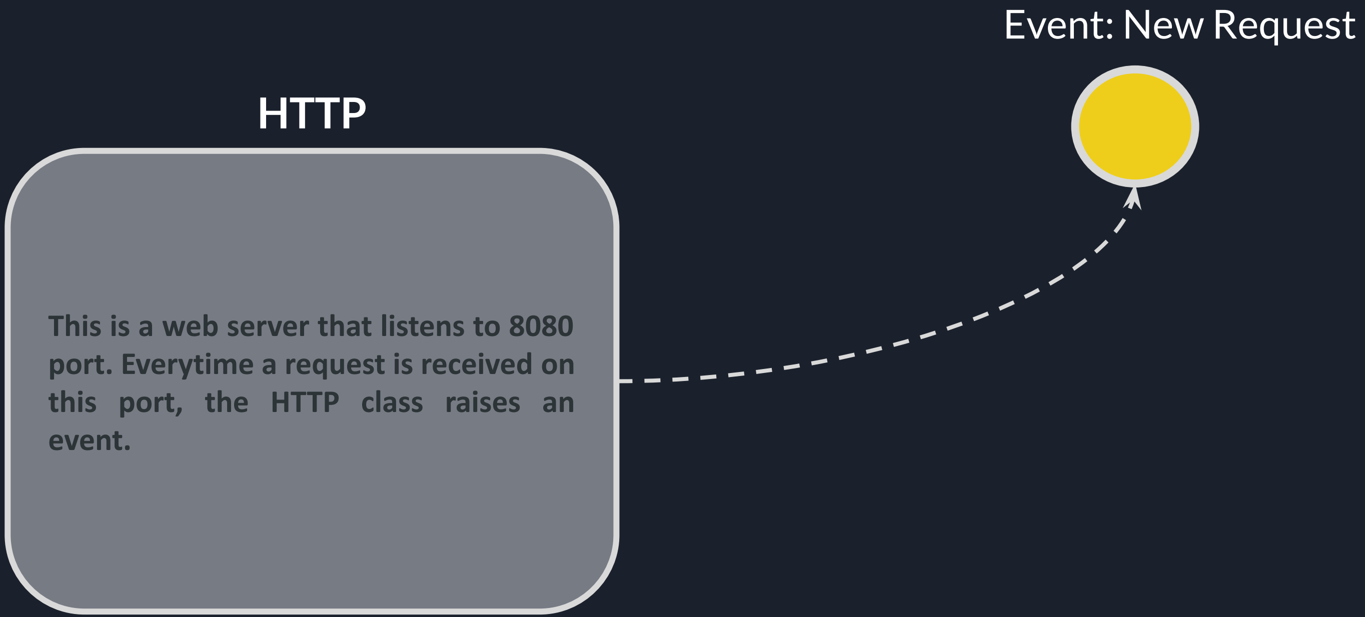
Events Module

Event is basically a signal that something has happened in our application.



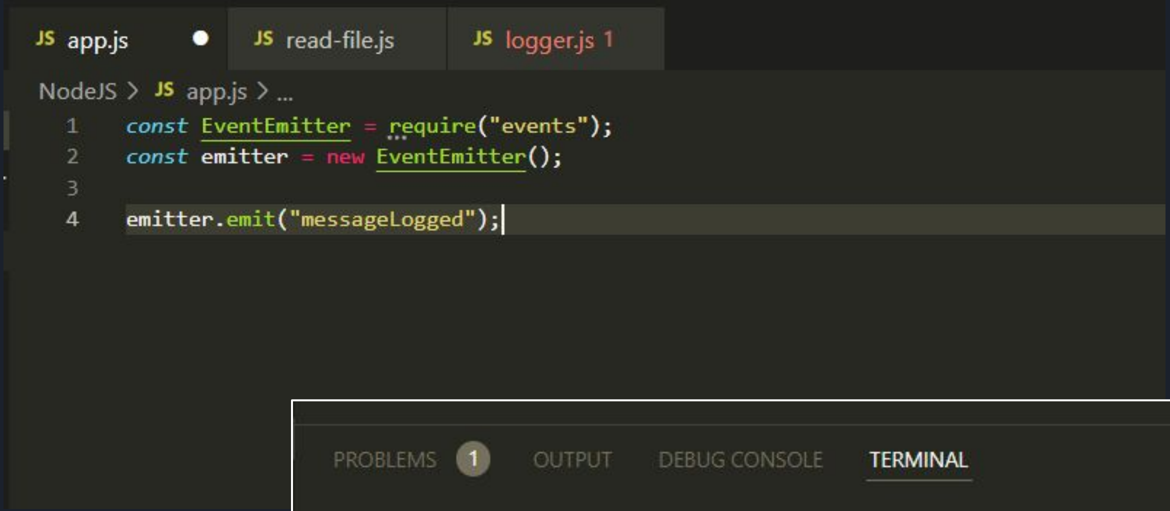
▼ Table of contents

URL -
https://nodejs.org/dist/latest-v16.x/docs/api/events.html

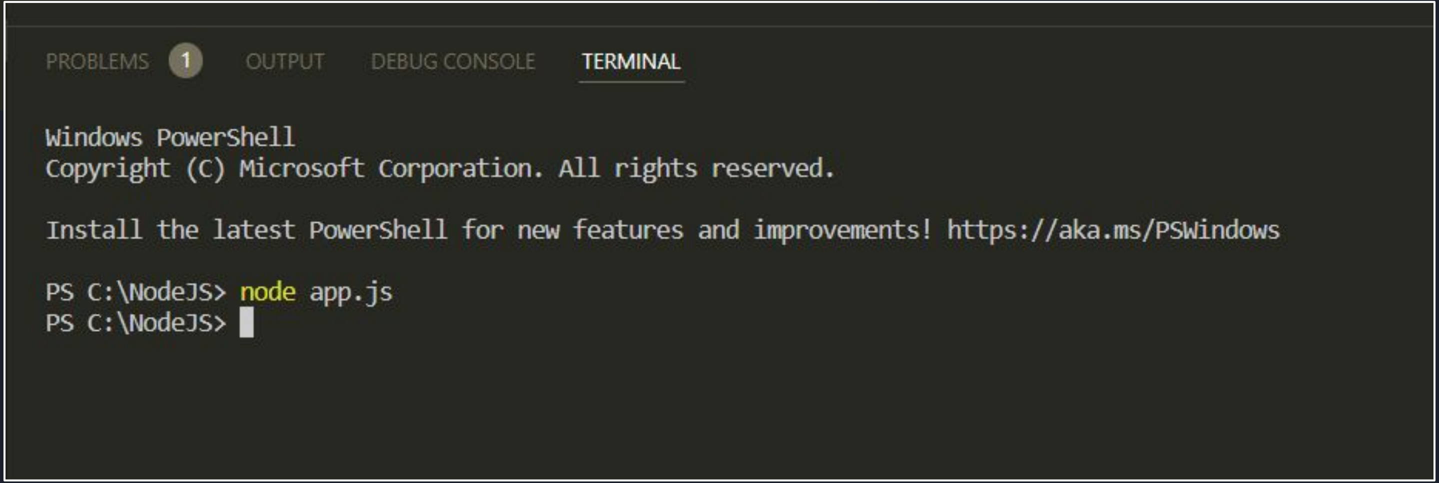


This is a web server that listens to 8080 port. Everytime a request is received on this port, the HTTP class raises an event.

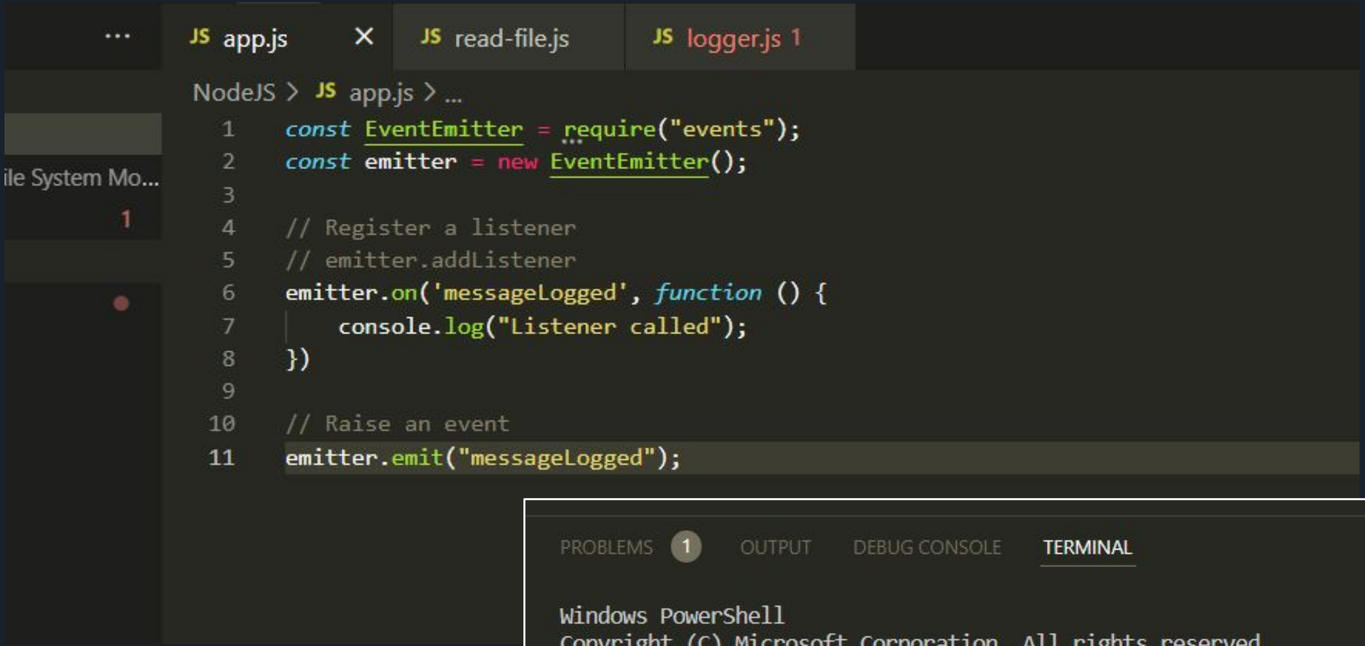
Our job here is to respond to the event, which basically involves reading that request and returning the right response.



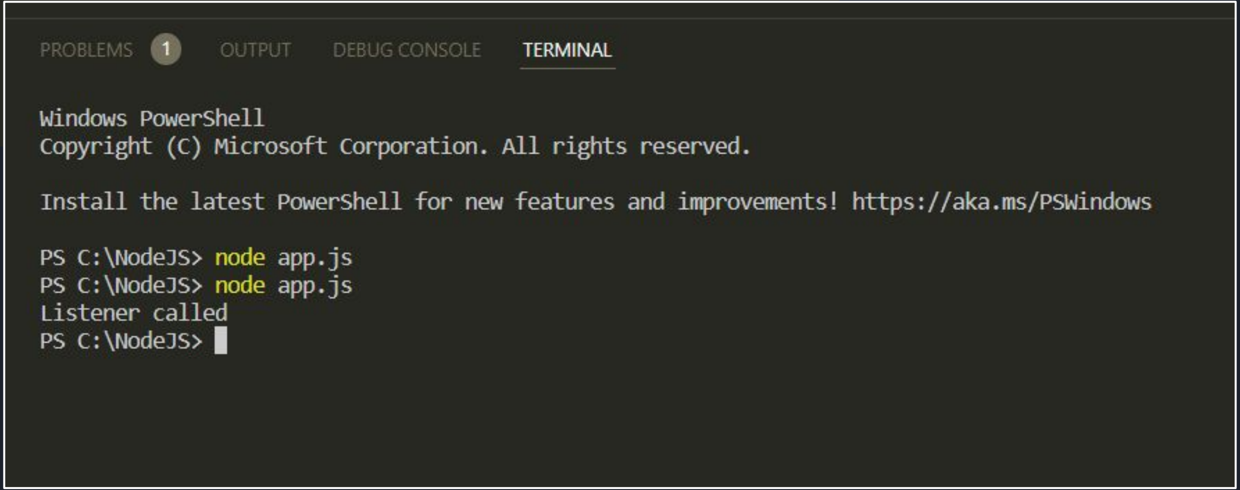
```javascript
const EventEmitter = require("events");
const emitter = new EventEmitter();

emitter.emit("messageLogged");
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\NodeJS> node app.js
PS C:\NodeJS>
```

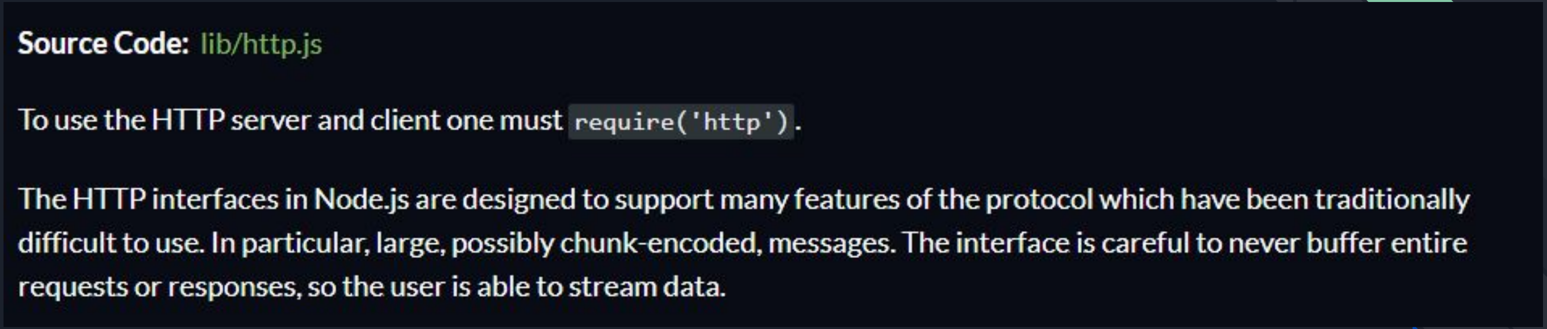


```javascript
const EventEmitter = require("events");
const emitter = new EventEmitter();

// Register a listener
// emitter.addListener
emitter.on('messageLogged', function () {
    console.log("Listener called");
})

// Raise an event
emitter.emit("messageLogged");
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\NodeJS> node app.js
PS C:\NodeJS> node app.js
Listener called
PS C:\NodeJS>
```

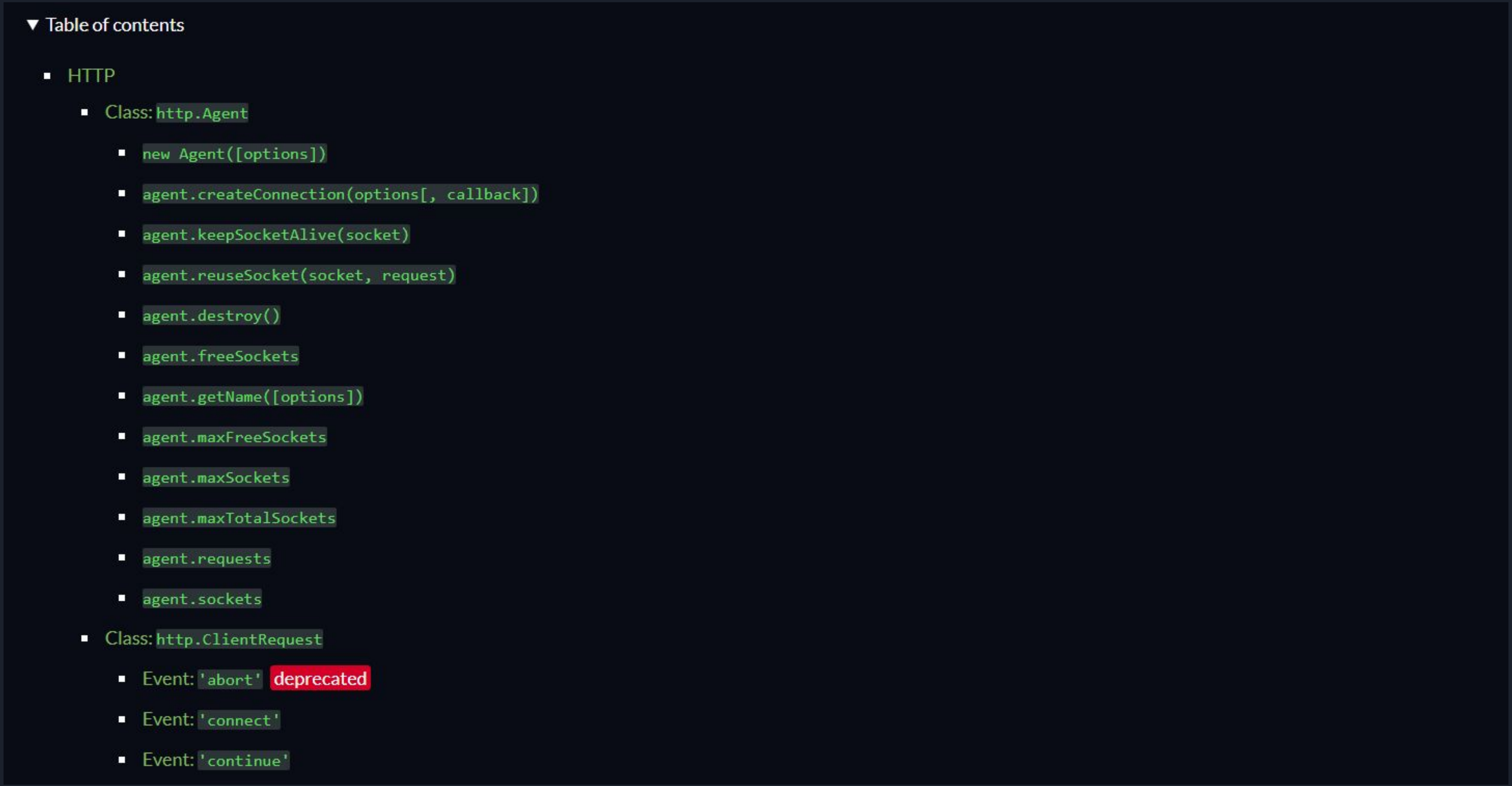
HTTP Module



Source Code: lib/http.js

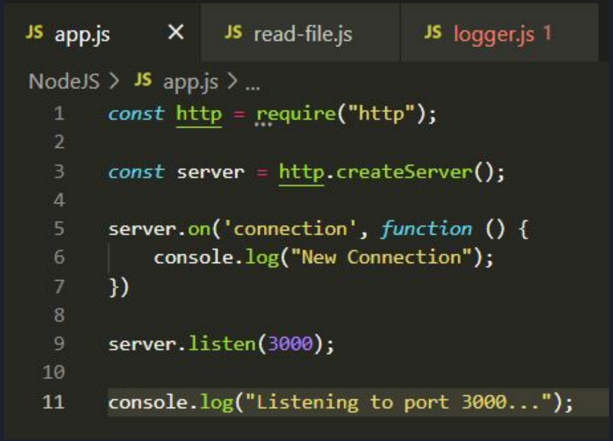
To use the HTTP server and client one must `require('http')`.

The HTTP interfaces in Node.js are designed to support many features of the protocol which have been traditionally difficult to use. In particular, large, possibly chunk-encoded, messages. The interface is careful to never buffer entire requests or responses, so the user is able to stream data.

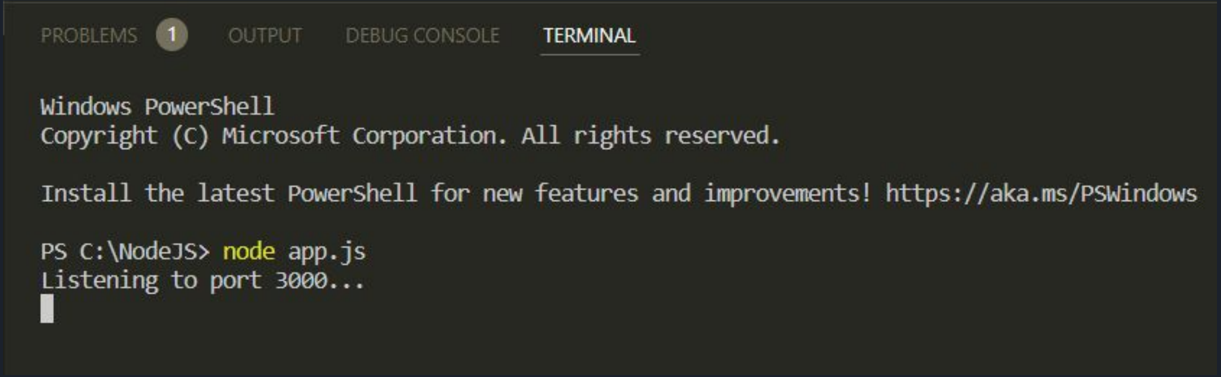


▼ Table of contents

URL - https://nodejs.org/dist/latest-v16.x/docs/api/http.html

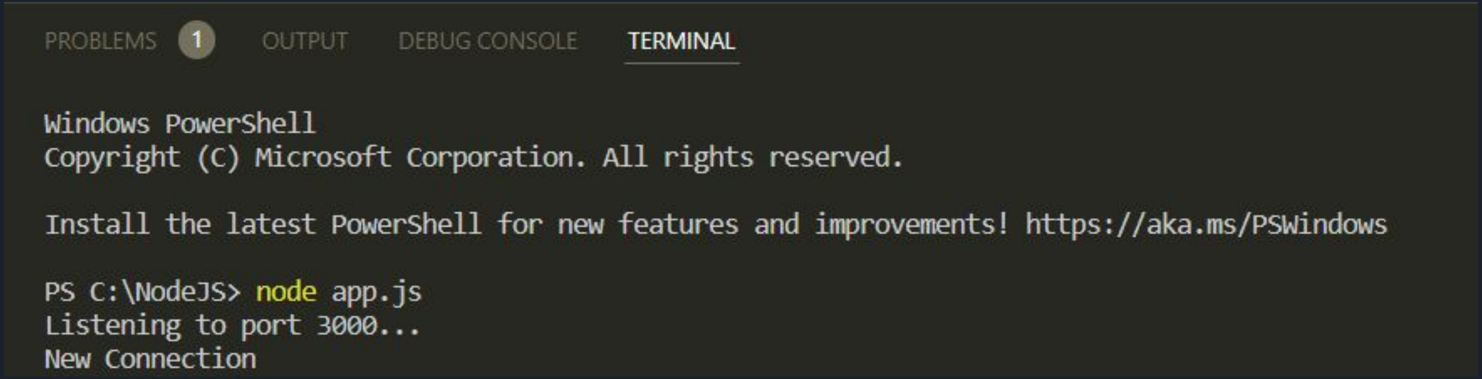


```js
const http = require("http");

const server = http.createServer();

server.on('connection', function () {
    console.log("New Connection");
})

server.listen(3000);

console.log("Listening to port 3000...");
```



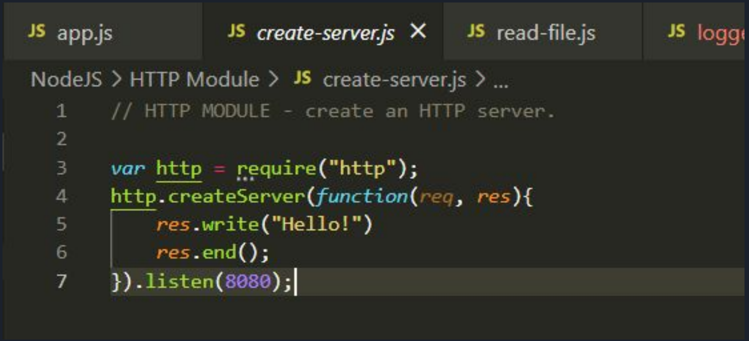
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\NodeJS> node app.js
Listening to port 3000...
```

After heading over to http://localhost:3000 on the browser:

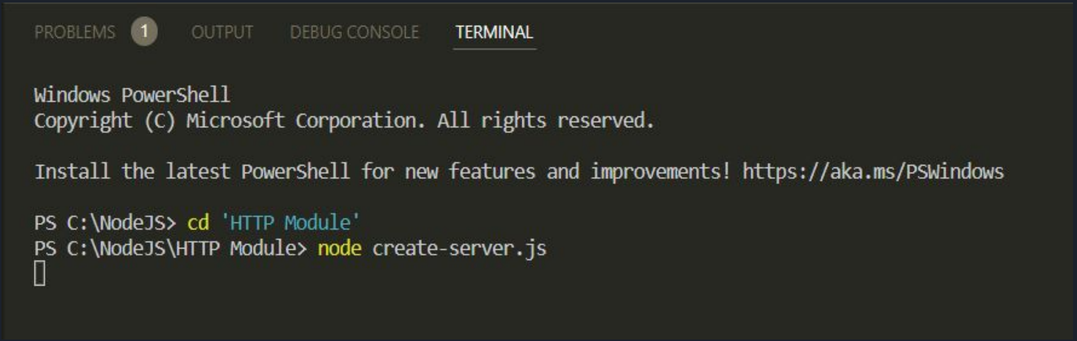


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\NodeJS> node app.js
Listening to port 3000...
New Connection
```

Creating a Server

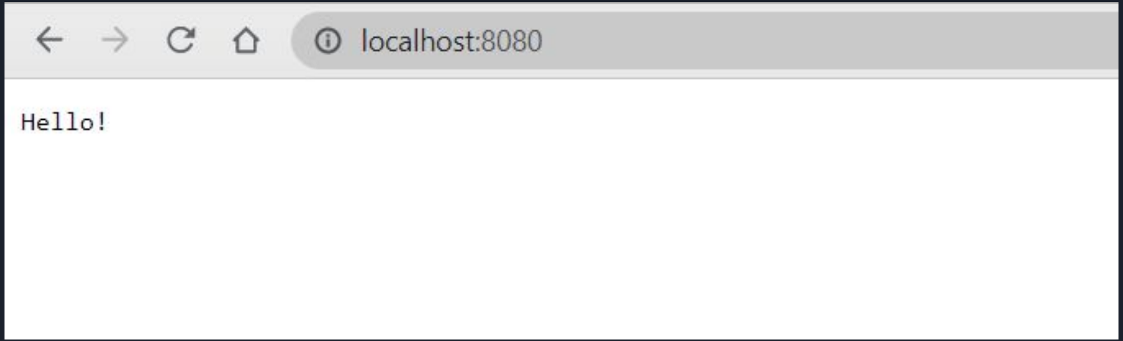


```javascript
// HTTP MODULE - create an HTTP server.

var http = require("http");
http.createServer(function(req, res){
    res.write("Hello!")
    res.end();
}).listen(8080);
```



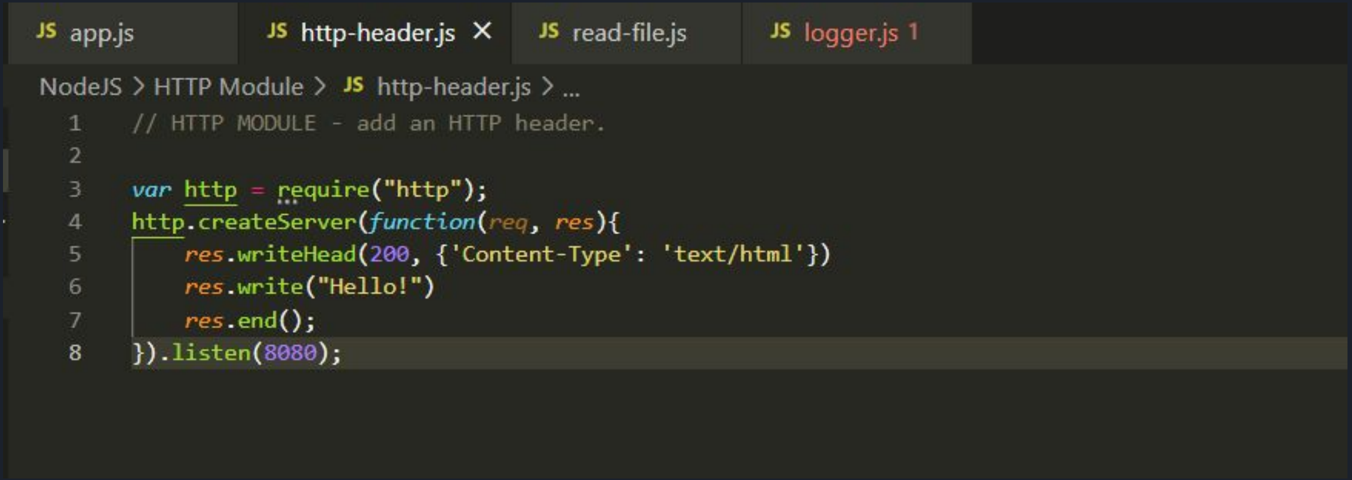
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\NodeJS> cd 'HTTP Module'
PS C:\NodeJS\HTTP Module> node create-server.js
```

After heading over to http://localhost:8080 on the browser:



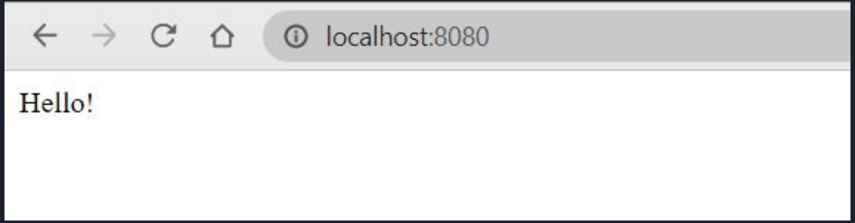
```
Hello!
```

Adding an HTTP header



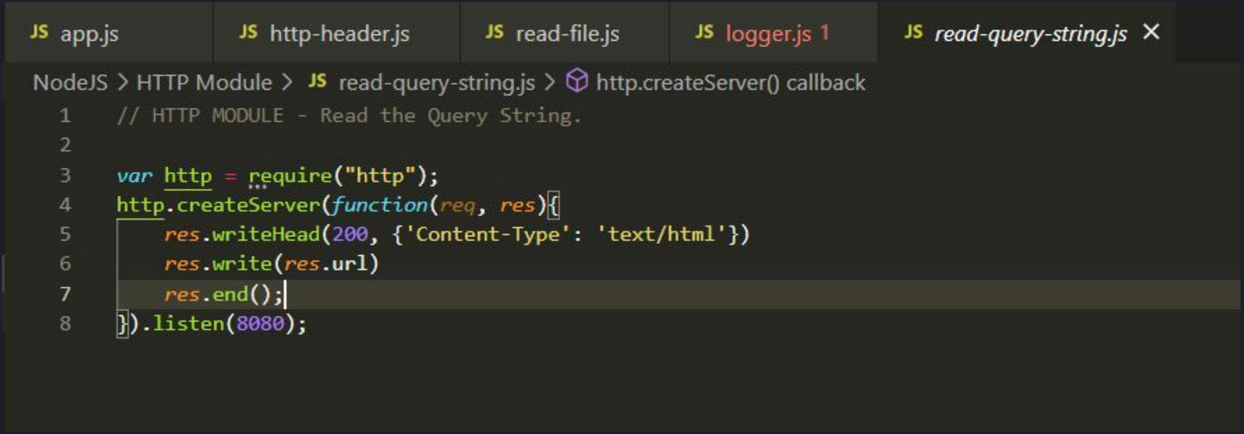
```
// HTTP MODULE - add an HTTP header.

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

After heading over to http://localhost:8080 on the browser:



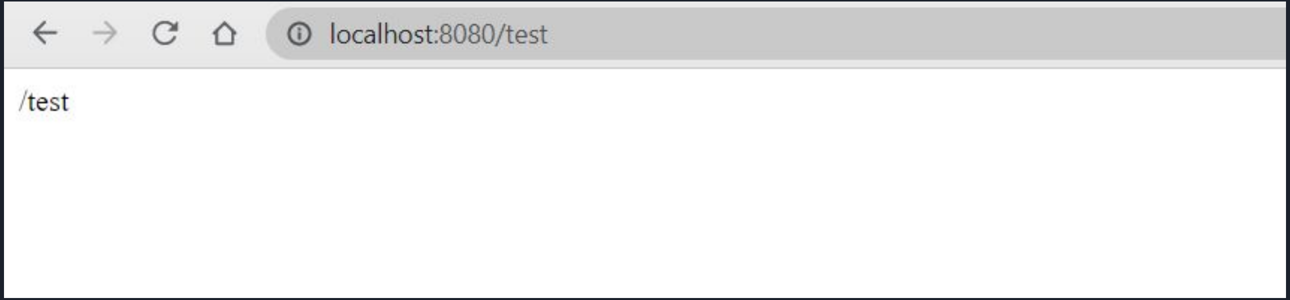
```
Hello!
```

Reading query string


JS app.js
JS http-header.js
JS read-file.js
JS logger.js 1
JS read-query-string.js ×
NodeJS > HTTP Module > JS read-query-string.js > http.createServer() callback

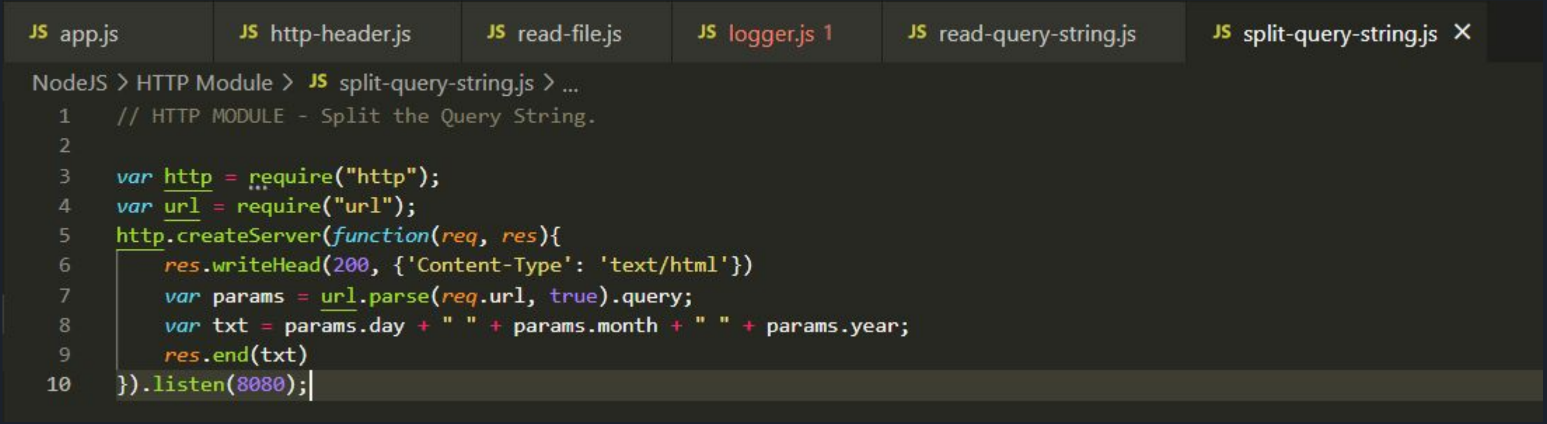
```
// HTTP MODULE - Read the Query String.

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

After heading over to http://localhost:8080 on the browser:



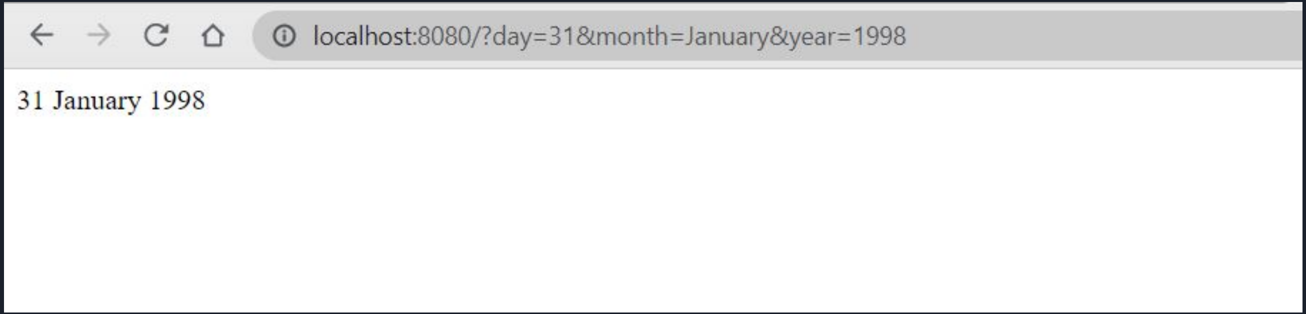
```
/test
```

Splitting the query string



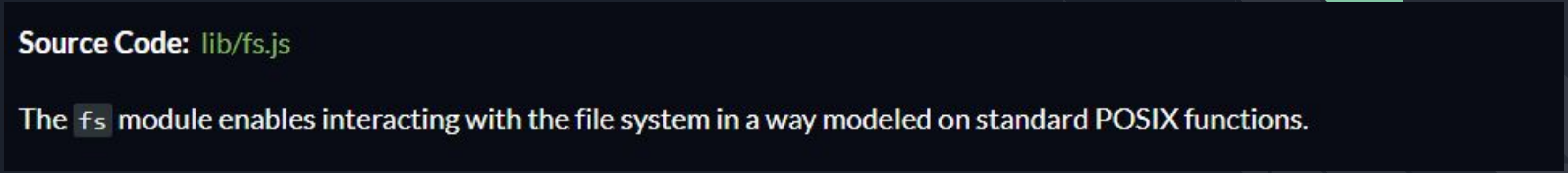
```
// HTTP MODULE - Split the Query String.

var http = require("http");
var url = require("url");
http.createServer(function(req, res){
    res.writeHead(200, {'Content-Type': 'text/html'})
    var params = url.parse(req.url, true).query;
    var txt = params.day + " " + params.month + " " + params.year;
    res.end(txt)
}).listen(8080);
```

After heading over to http://localhost:8080 on the browser:

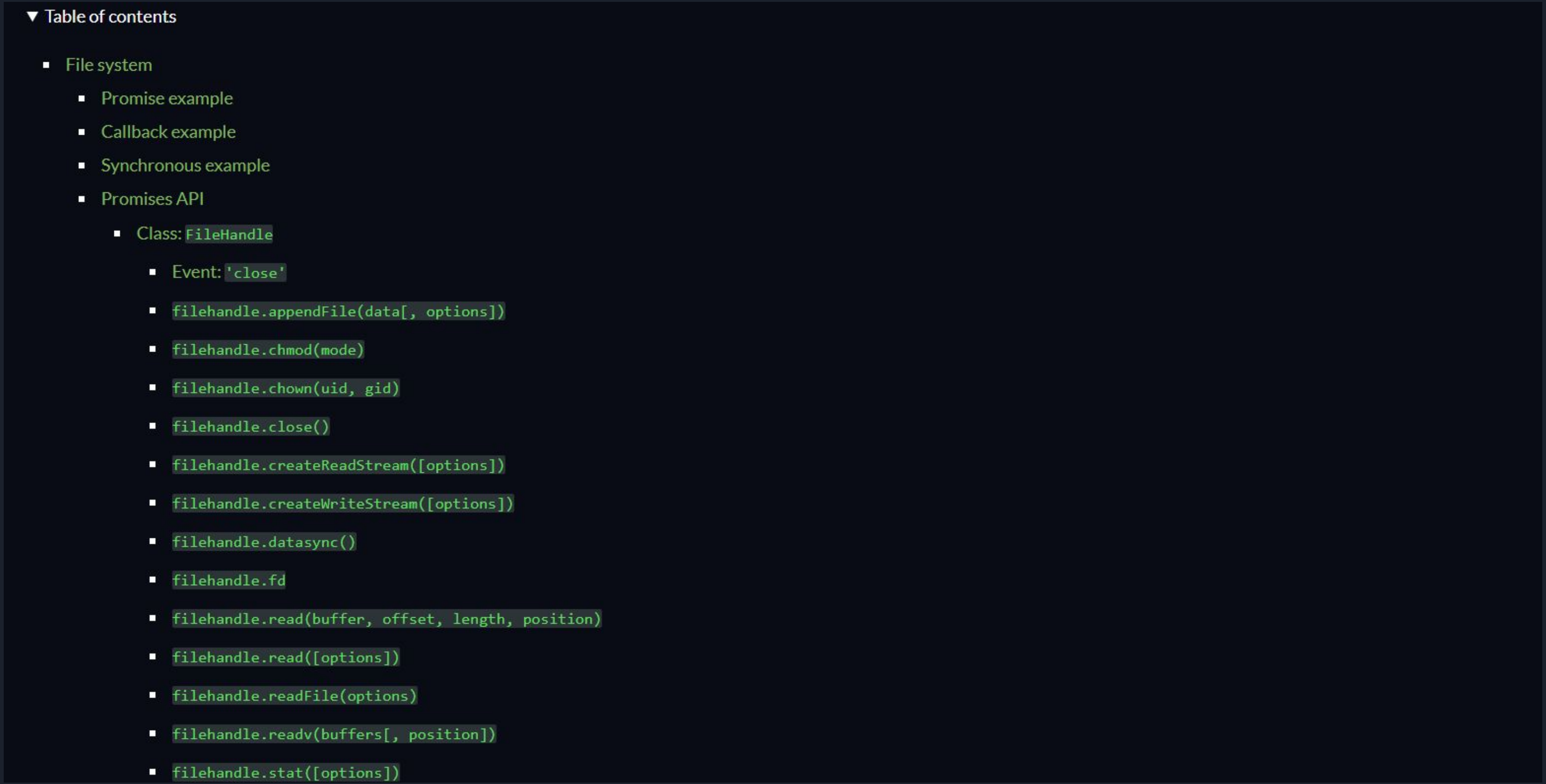


```
31 January 1998
```

File System Module

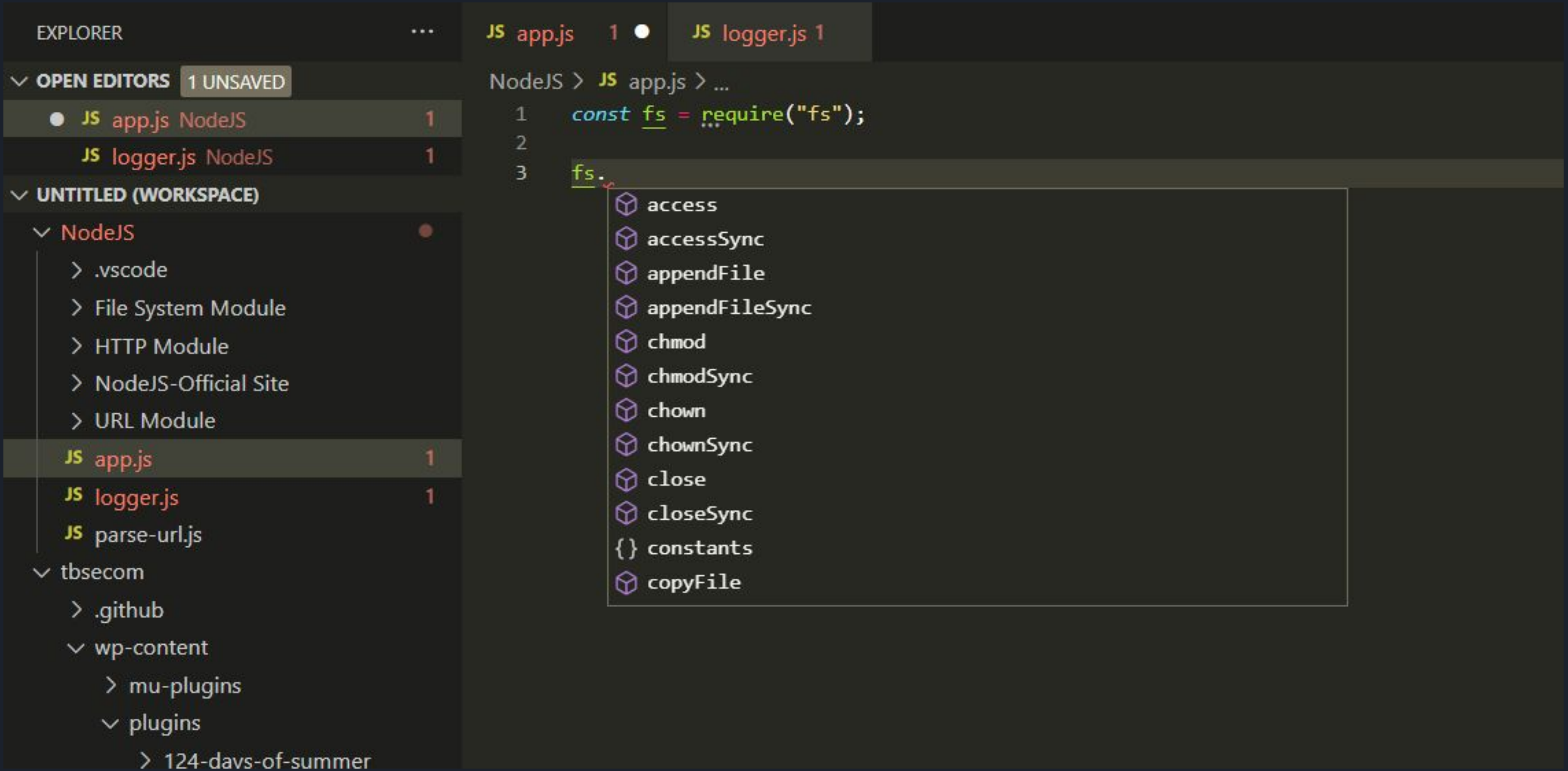

Source Code: lib/fs.js

The `fs` module enables interacting with the file system in a way modeled on standard POSIX functions.



▼ Table of contents

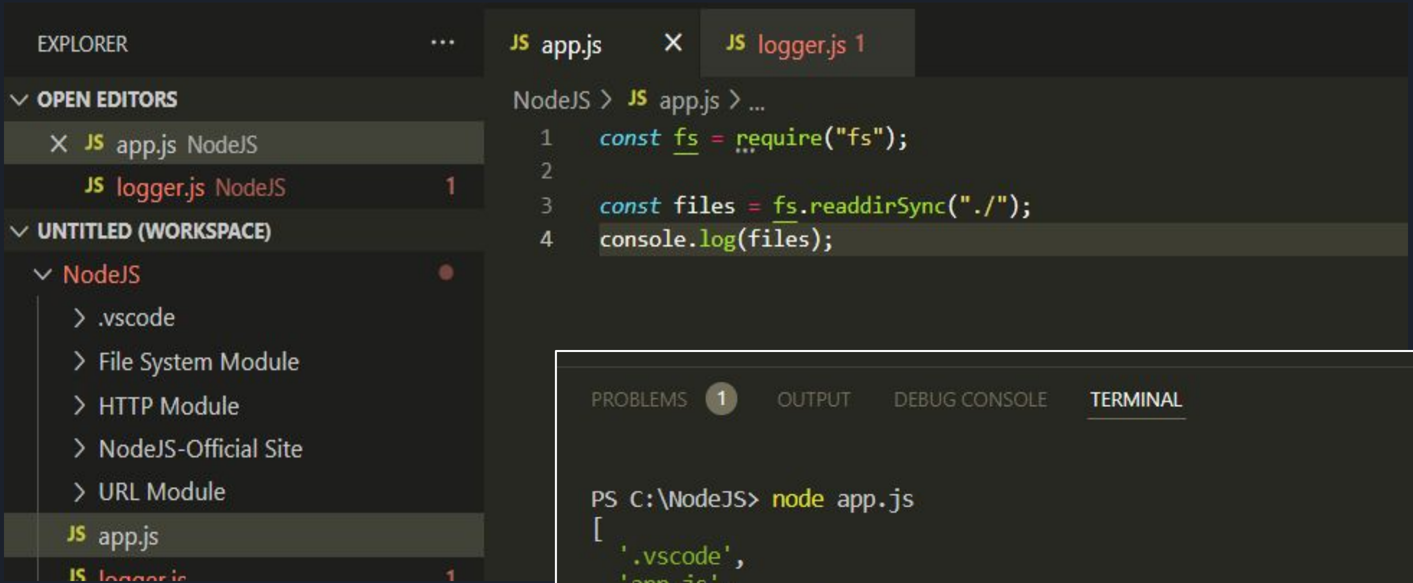
URL - https://nodejs.org/dist/latest-v16.x/docs/api/os.html



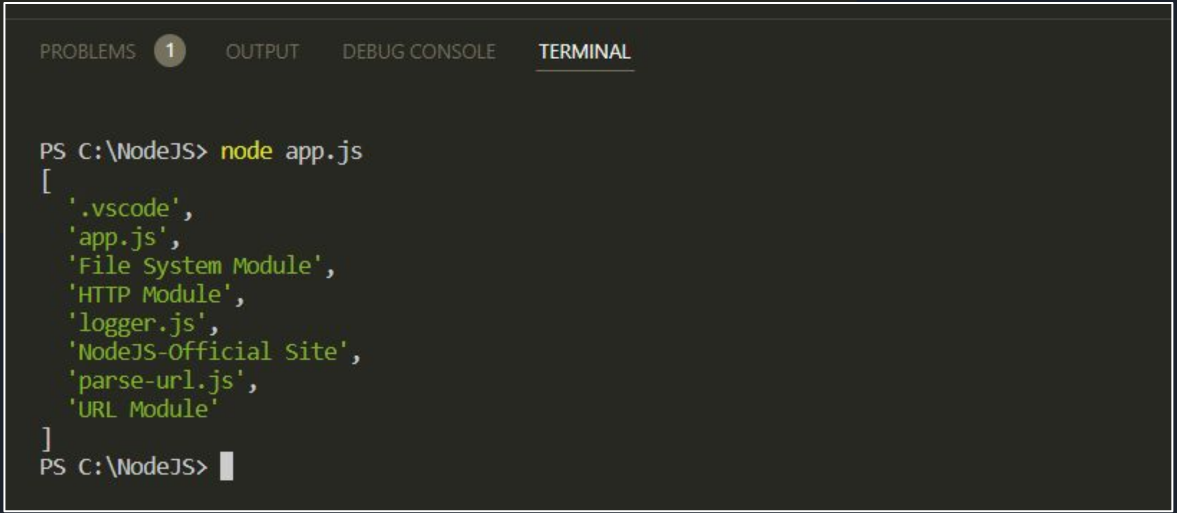
```
const fs = require("fs");

fs.
```

```
access
accessSync
appendFile
appendFileSync
chmod
chmodSync
chown
chownSync
close
closeSync
constants
copyFile
```

Reading contents of a directory - sync

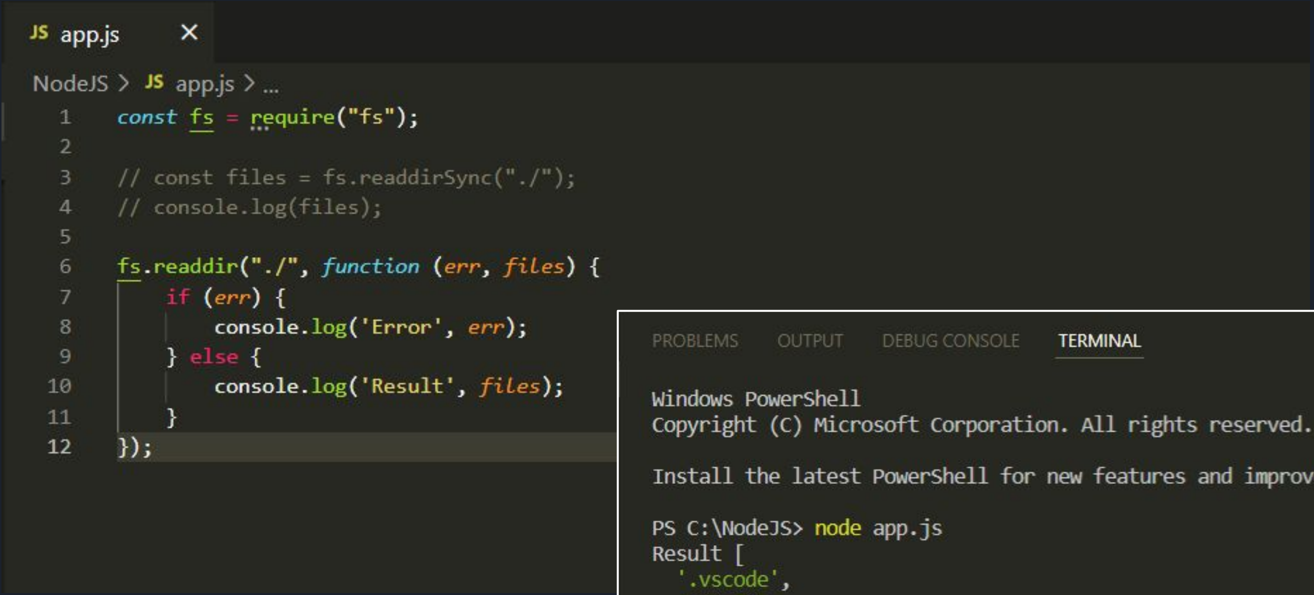


```javascript
const fs = require("fs");

const files = fs.readdirSync("./");
console.log(files);
```



```
PS C:\NodeJS> node app.js
[
  '.vscode',
  'app.js',
  'File System Module',
  'HTTP Module',
  'logger.js',
  'NodeJS-Official Site',
  'parse-url.js',
  'URL Module'
]
PS C:\NodeJS>
```

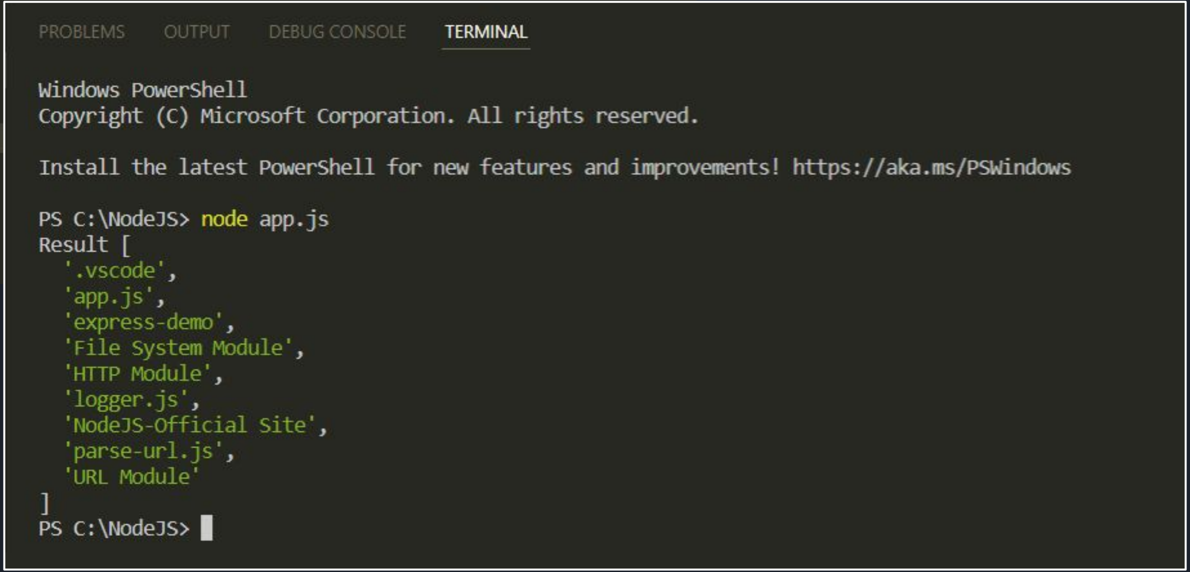
Reading contents of a directory - async



app.js ×

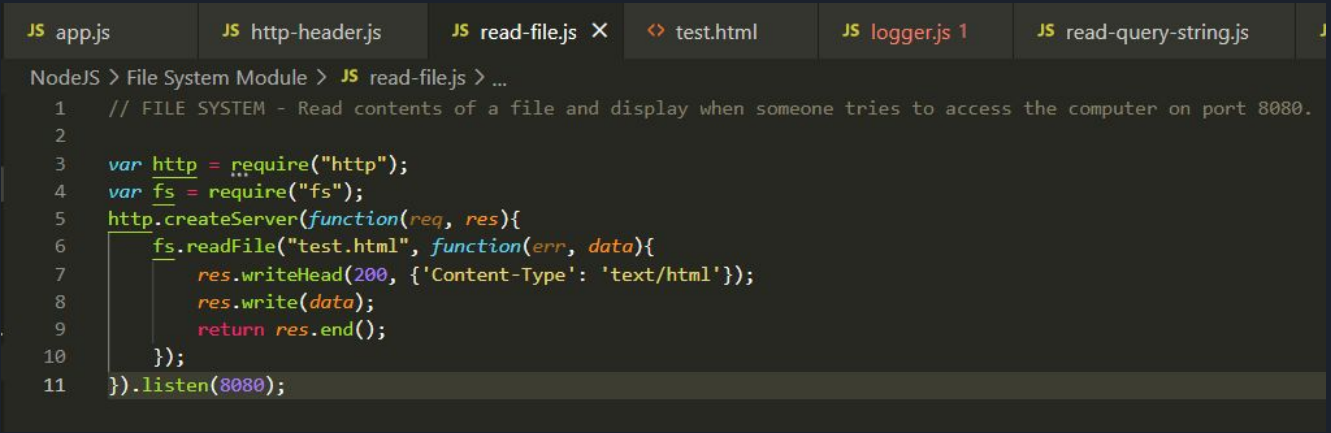
NodeJS > app.js > ...

```javascript
const fs = require("fs");

// const files = fs.readdirSync("./");
// console.log(files);

fs.readdir("./", function (err, files) {
    if (err) {
        console.log('Error', err);
    } else {
        console.log('Result', files);
    }
});
```



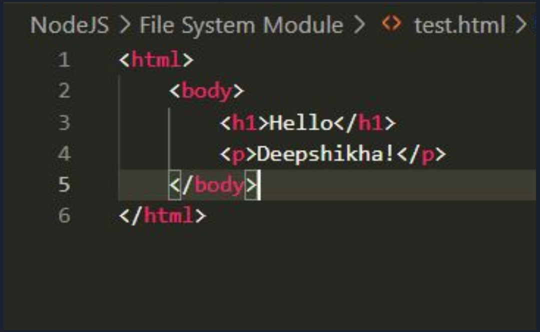
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\NodeJS> node app.js
Result [
  '.vscode',
  'app.js',
  'express-demo',
  'File System Module',
  'HTTP Module',
  'logger.js',
  'NodeJS-Official Site',
  'parse-url.js',
  'URL Module'
]
PS C:\NodeJS> 
```

Reading a file



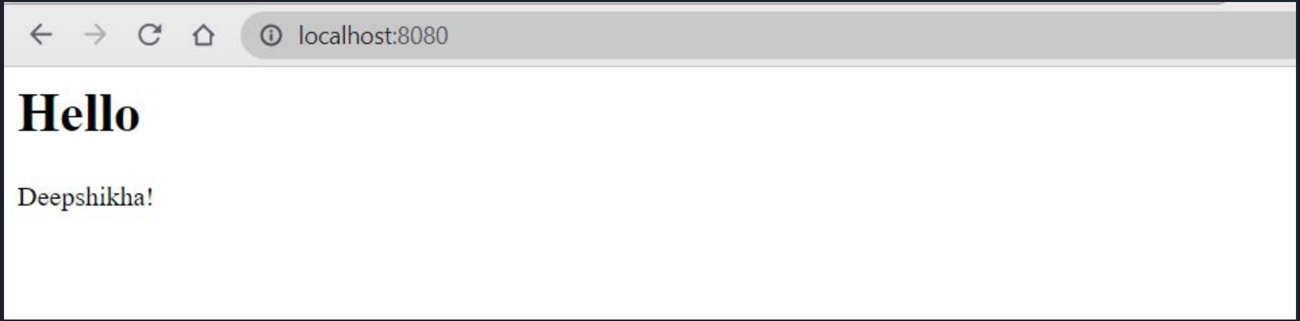
```javascript
// FILE SYSTEM - Read contents of a file and display when someone tries to access the computer on port 8080.

var http = require("http");
var fs = require("fs");
http.createServer(function(req, res){
    fs.readFile("test.html", function(err, data){
        res.writeHead(200, {'Content-Type': 'text/html'});
        res.write(data);
        return res.end();
    });
}).listen(8080);
```

test.html



```html
<html>
    <body>
        <h1>Hello</h1>
        <p>Deepshikha!</p>
    </body>
</html>
```

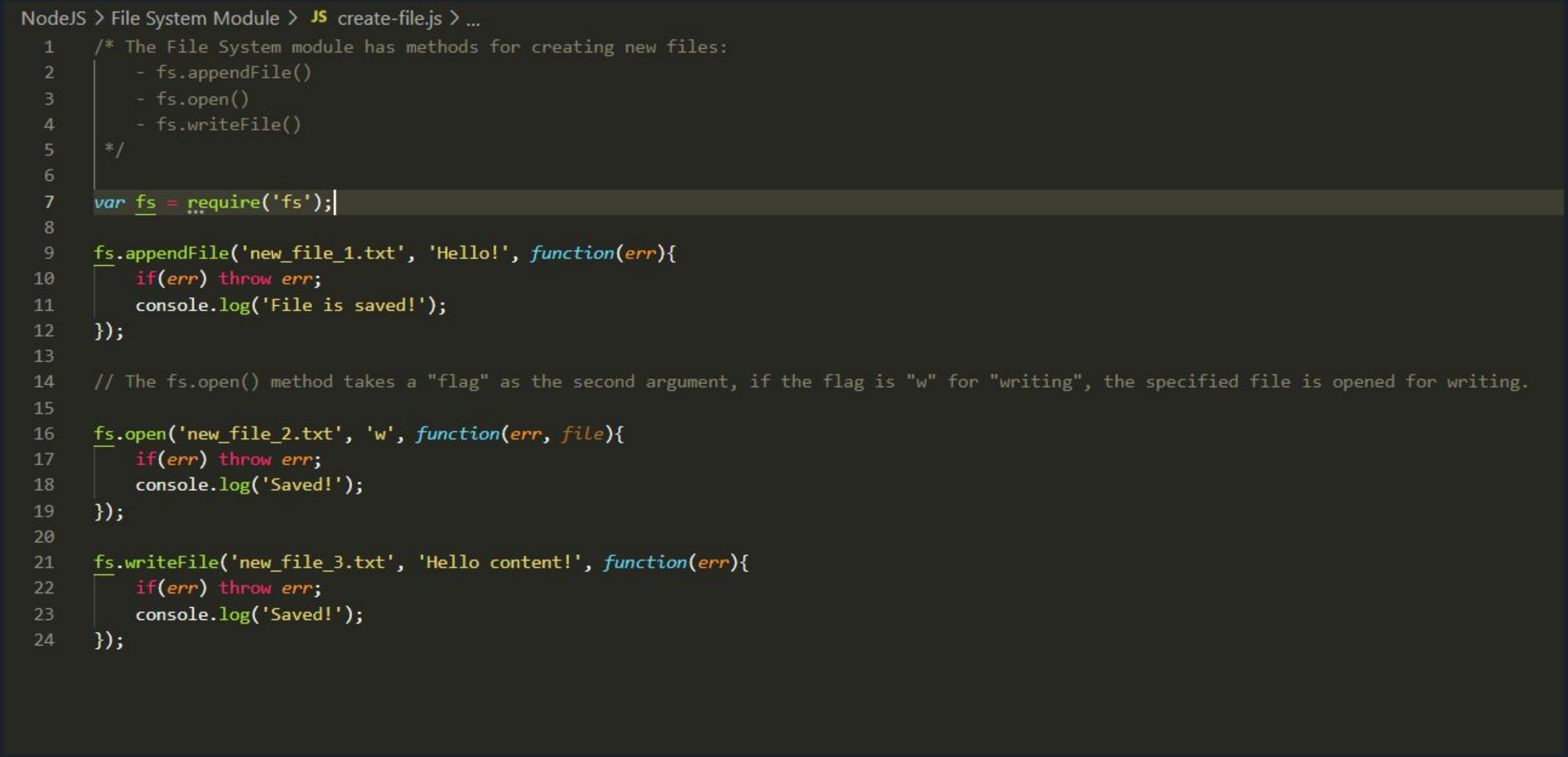
After heading over to http://localhost:8080 on the browser:



Hello

Deepshikha!

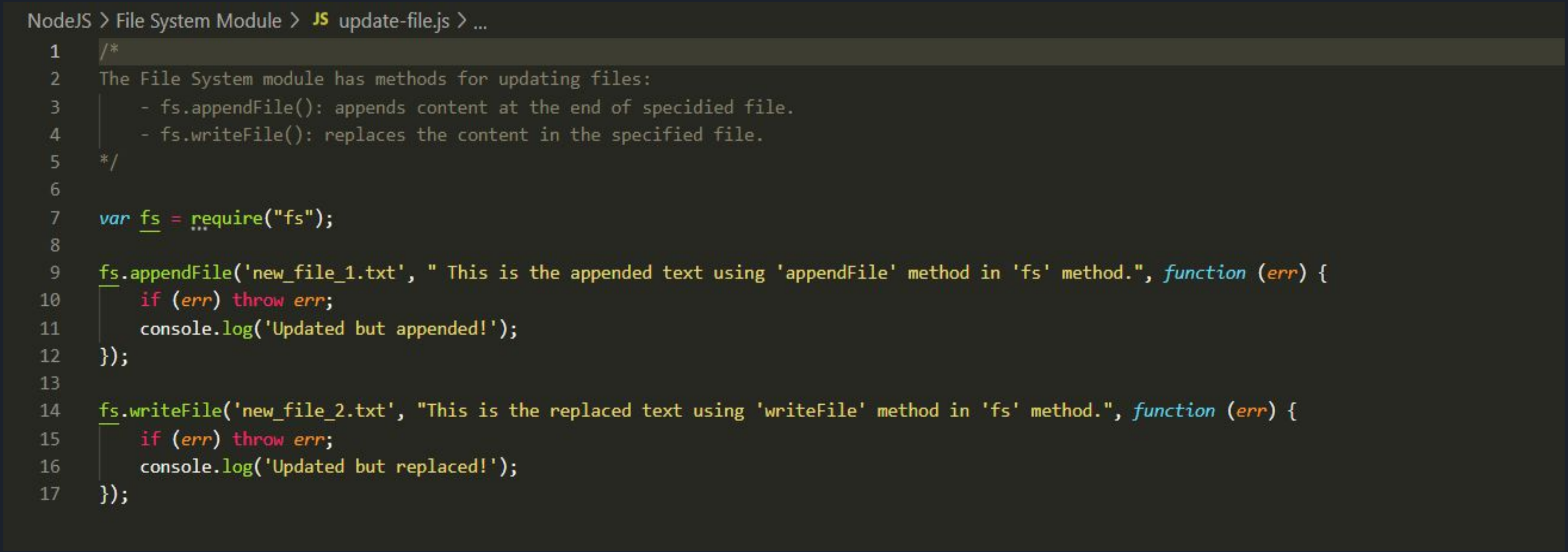
Creating a file



NodeJS > File System Module > JS create-file.js > ...

```
/* The File System module has methods for creating new files:
    - fs.appendFile()
    - fs.open()
    - fs.writeFile()
 */

var fs = require('fs');

fs.appendFile('new_file_1.txt', 'Hello!', function(err){
    if(err) throw err;
    console.log('File is saved!');
});

// The fs.open() method takes a "flag" as the second argument, if the flag is "w" for "writing", the specified file is opened for writing.

fs.open('new_file_2.txt', 'w', function(err, file){
    if(err) throw err;
    console.log('Saved!');
});

fs.writeFile('new_file_3.txt', 'Hello content!', function(err){
    if(err) throw err;
    console.log('Saved!');
});
```

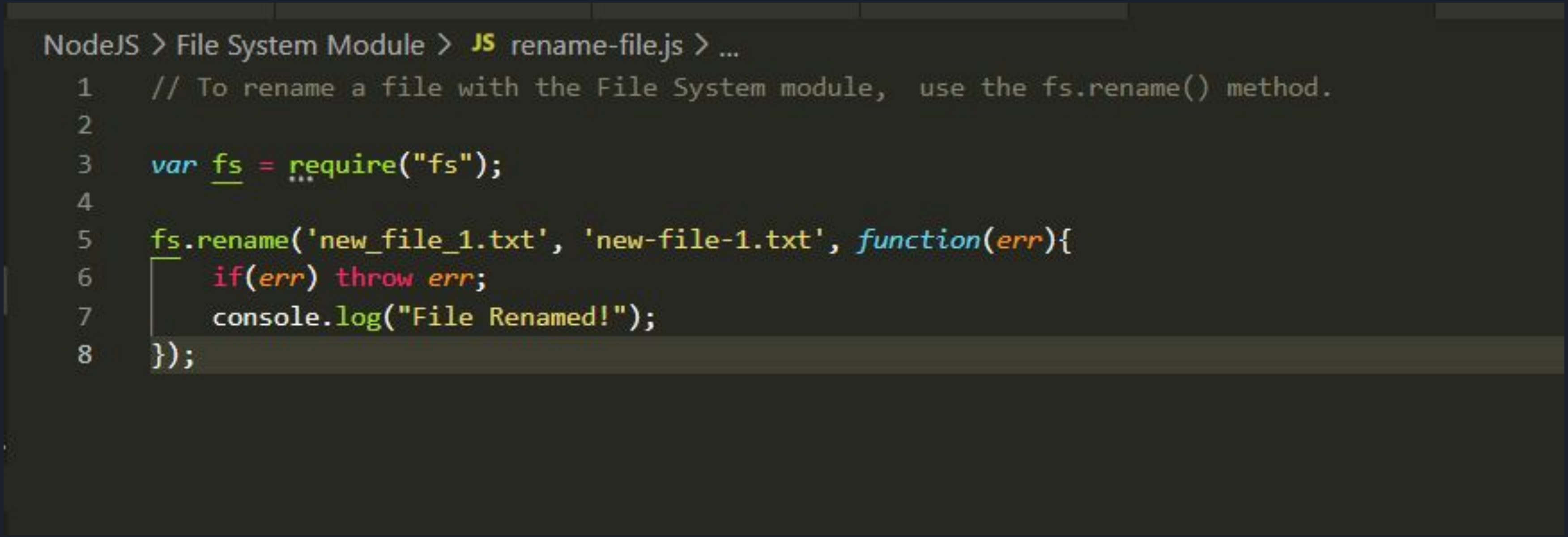
Updating a file



NodeJS > File System Module > JS update-file.js > ...

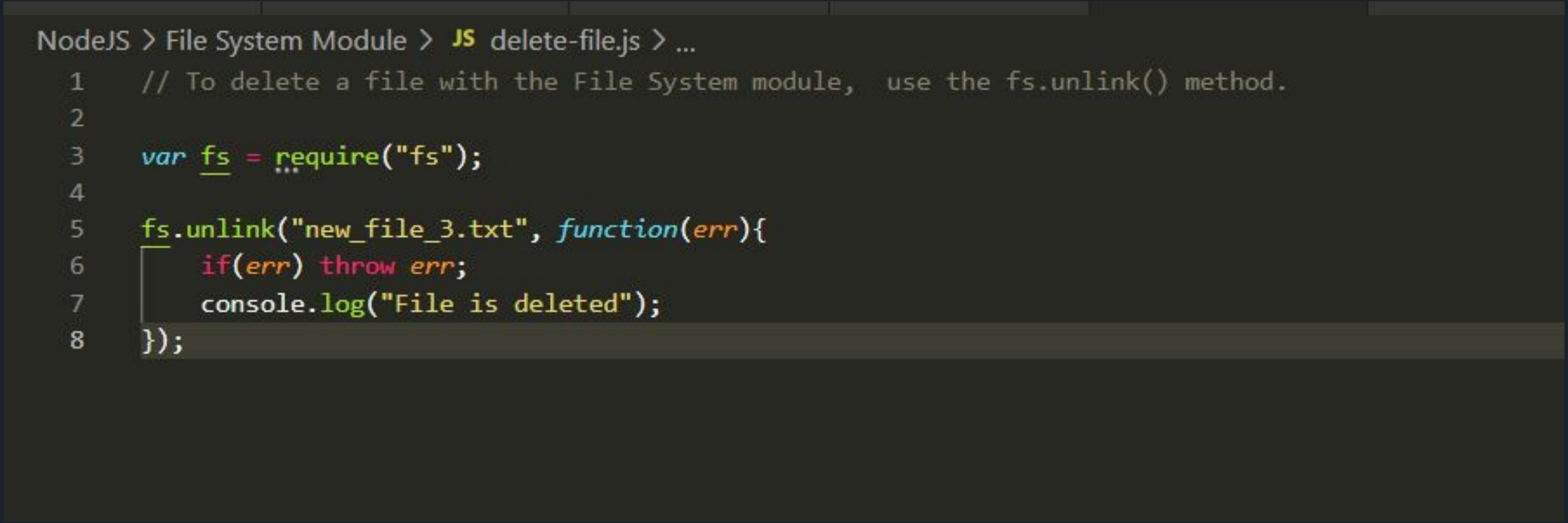
```
/*
The File System module has methods for updating files:
    - fs.appendFile(): appends content at the end of specidied file.
    - fs.writeFile(): replaces the content in the specified file.
*/

var fs = require("fs");

fs.appendFile('new_file_1.txt', " This is the appended text using 'appendFile' method in 'fs' method.", function (err) {
    if (err) throw err;
    console.log('Updated but appended!');
});

fs.writeFile('new_file_2.txt', "This is the replaced text using 'writeFile' method in 'fs' method.", function (err) {
    if (err) throw err;
    console.log('Updated but replaced!');
});
```

Rename a file


NodeJS > File System Module > JS rename-file.js > ...

```
// To rename a file with the File System module,  use the fs.rename() method.

var fs = require("fs");

fs.rename('new_file_1.txt', 'new-file-1.txt', function(err){
    if(err) throw err;
    console.log("File Renamed!");
});
```

Deleting a file



NodeJS > File System Module > JS delete-file.js > ...

```javascript
// To delete a file with the File System module,  use the fs.unlink() method.

var fs = require("fs");

fs.unlink("new_file_3.txt", function(err){
    if(err) throw err;
    console.log("File is deleted");
});
```

Other common modules

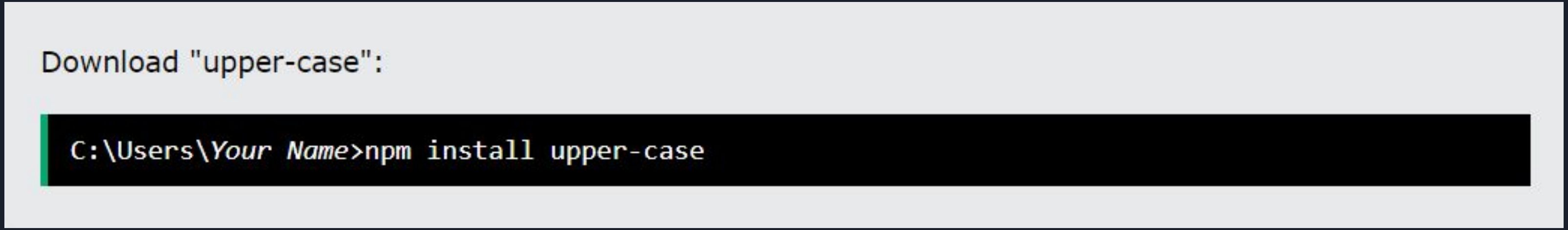
- URL
- Events
- Nodemailer Module
- MySQL
- MongoDB

Node.js NPM

- NPM is a package manager for Node.js packages, or modules if you like.
- www.npmjs.com hosts thousands of free packages to download and use.
- The NPM program is installed on your computer when you install Node.js

Download a Package -

Open the command line interface and tell NPM to download the package you want.



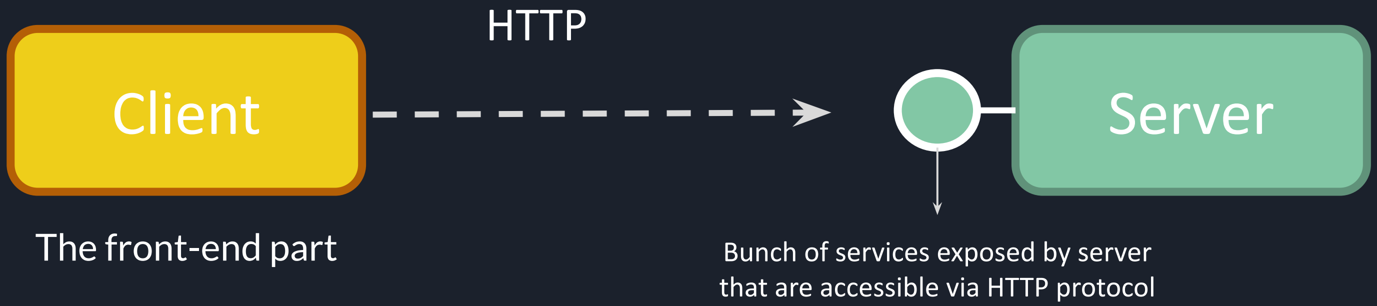
Download "upper-case":

```
C:\Users\Your Name>npm install upper-case
```

Express

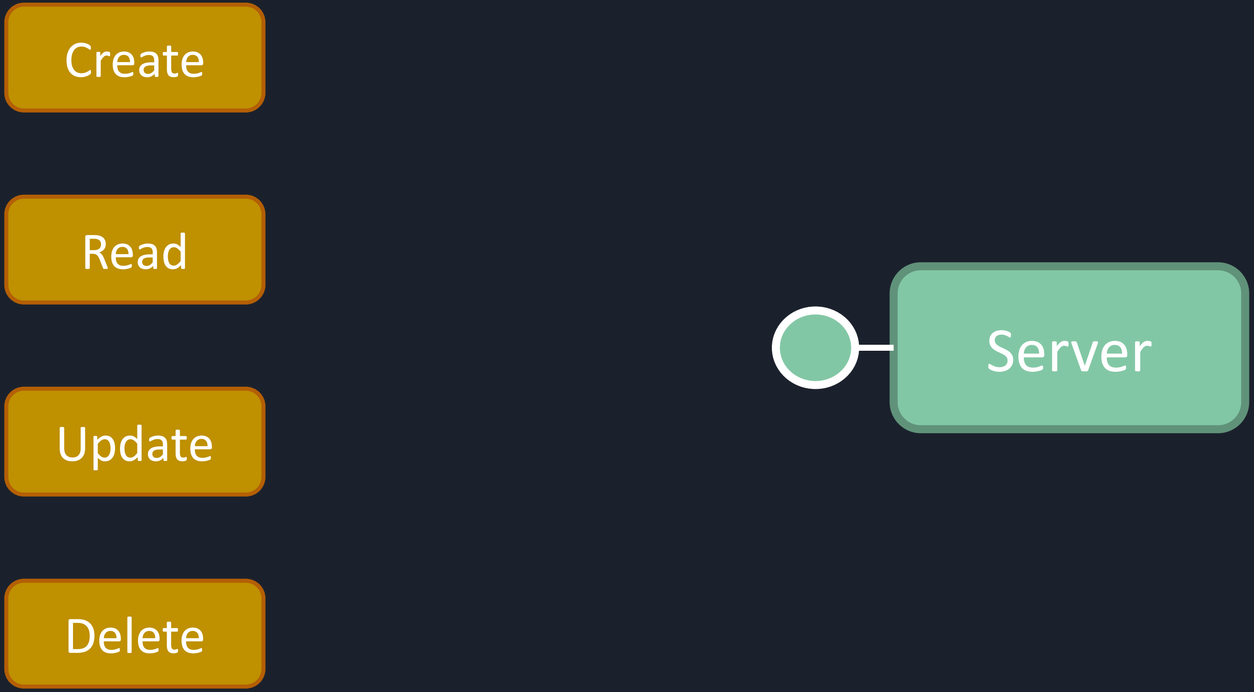
This is a fast and lightweight framework for building web applications.

Web Applications


HTTP
Client
Server
The front-end part
Bunch of services exposed by server that are accessible via HTTP protocol

- The communication between client and server happens using the HTTP protocol.
- The client can directly call these services by sending HTTP requests.

CRUD Operations



http://spotify.com/api/customers

The address can start from http or https. If you want to use a secure channel then use https.

Domain of the application

This is not compulsory, but this convention is usually followed to expose the restful services.

This refers to the collection of customers. In REST world, this part is called a **resource**.

HTTP METHODS

GET	For getting data
POST	For creating data
PUT	For updating data
DELETE	For deleting data

GET CUSTOMERS

Request

```
GET /api/customers
```

Indicates a list of customers

Response

```
[
        { id: 1, name: ‘abc’ },
        { id: 2, name: ‘def’ },
        ..
]
```

GET A CUSTOMER

Request

```
GET /api/customers/1
```

Response

```
{ id: 1, name: ‘abc’ }
```

CREATE A CUSTOMER

Request

```
POST /api/customers

{ name: ‘abc’ }
```

Response

```
{ id: 1, name: ‘abc’ }
```

UPDATE A CUSTOMER

Request

```
PUT /api/customers/1

{ name: 'abc1' }
```

Response

```
{ id: 1, name: 'abc1' }
```

DELETE A CUSTOMER

Request

```
DELETE /api/customers/1
```

Response

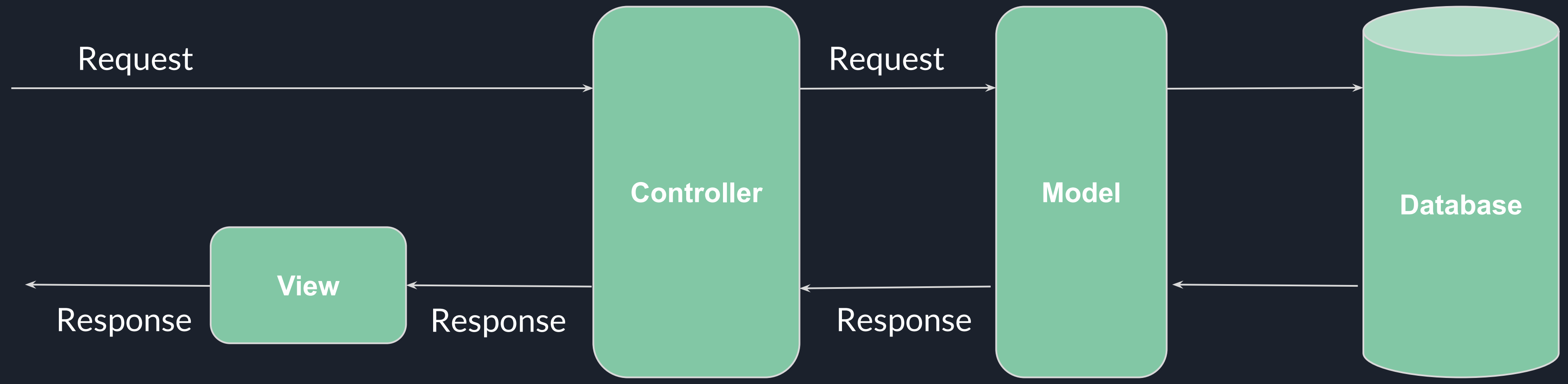
GET /api/customers
GET /api/customers/1
POST /api/customers
PUT /api/customers/1
DELETE /api/customers/1

MVC STRUCTURE

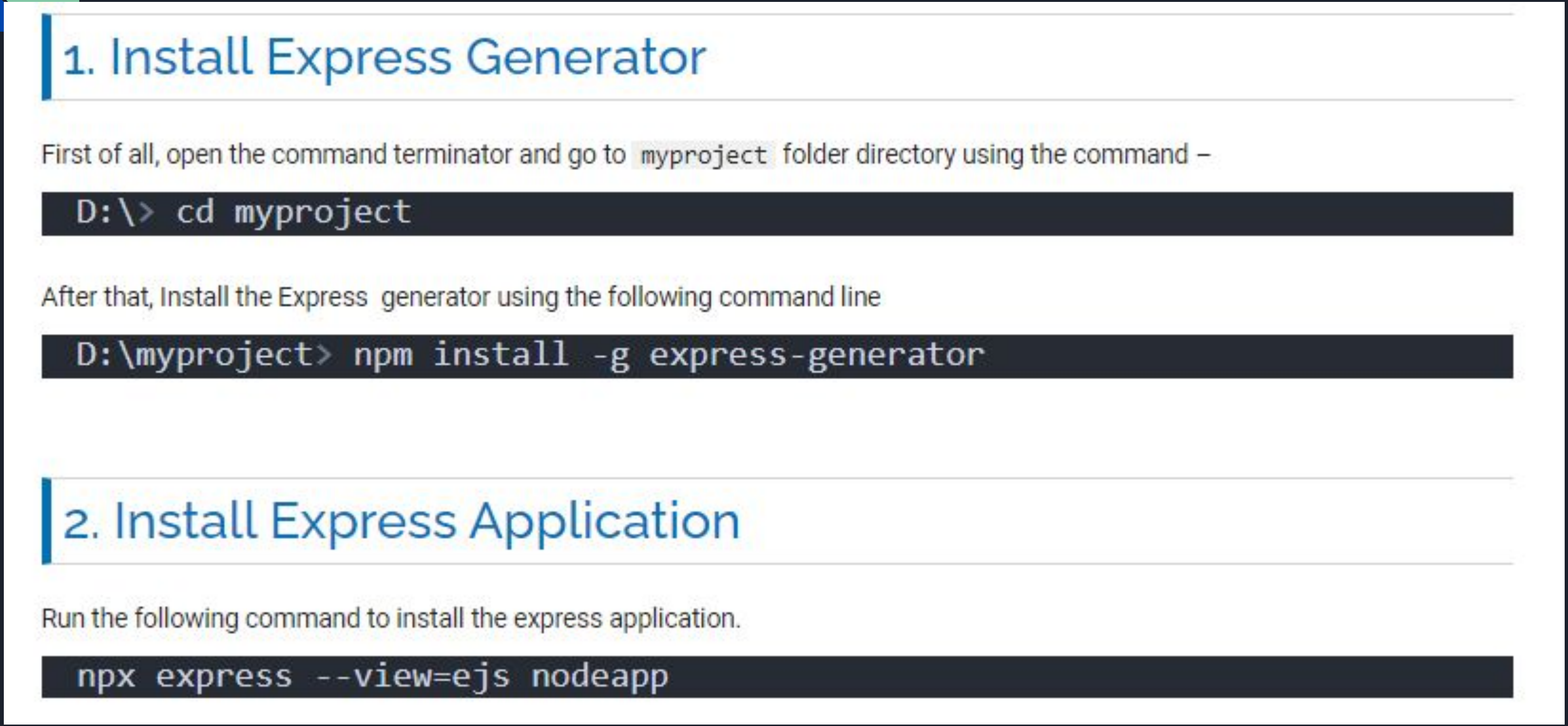
MVC is the most popular & useful structure for web application and it describes as -

MODEL	It can handle the database
VIEW	It can handle the client-side web pages
CONTROLLER	It can control the request & response of Model & View

STRUCTURE


Request
Controller
Request
Model
Database
View
Response
Response
Response

How to Install Express Application?



1. Install Express Generator

First of all, open the command terminator and go to `myproject` folder directory using the command –

```
D:\> cd myproject
```

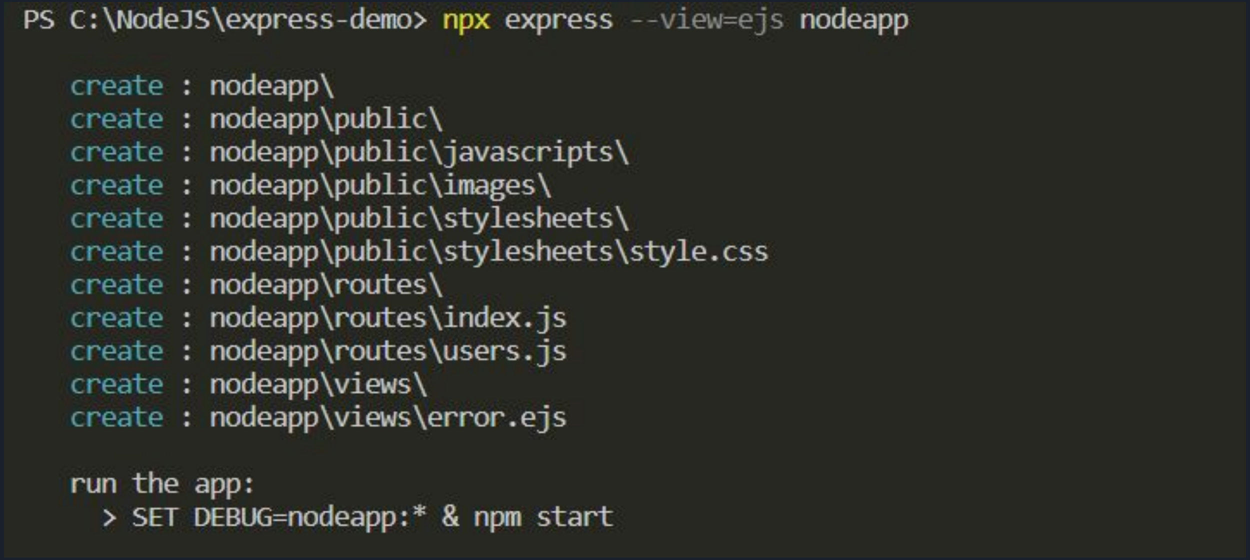
After that, Install the Express generator using the following command line

```
D:\myproject> npm install -g express-generator
```

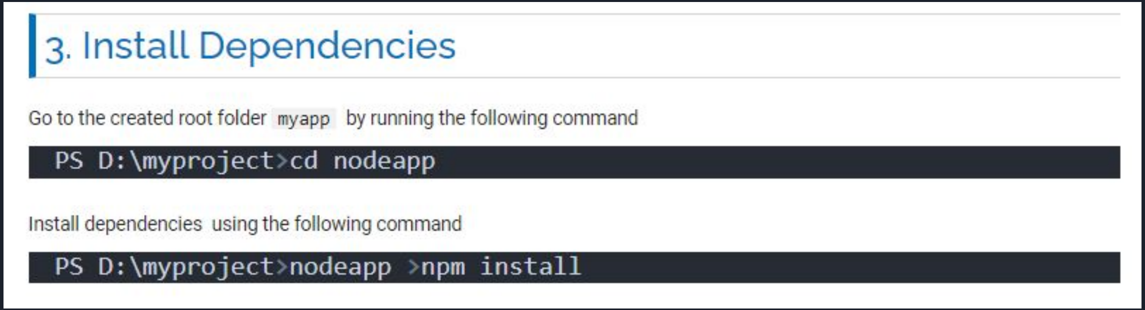
2. Install Express Application

Run the following command to install the express application.

```
npx express --view=ejs nodeapp
```



```
PS C:\NodeJS\express-demo> npx express --view=ejs nodeapp

   create : nodeapp\
   create : nodeapp\public\
   create : nodeapp\public\javascripts\
   create : nodeapp\public\images\
   create : nodeapp\public\stylesheets\
   create : nodeapp\public\stylesheets\style.css
   create : nodeapp\routes\
   create : nodeapp\routes\index.js
   create : nodeapp\routes\users.js
   create : nodeapp\views\
   create : nodeapp\views\error.ejs

   run the app:
     > SET DEBUG=nodeapp:* & npm start
```



3. Install Dependencies

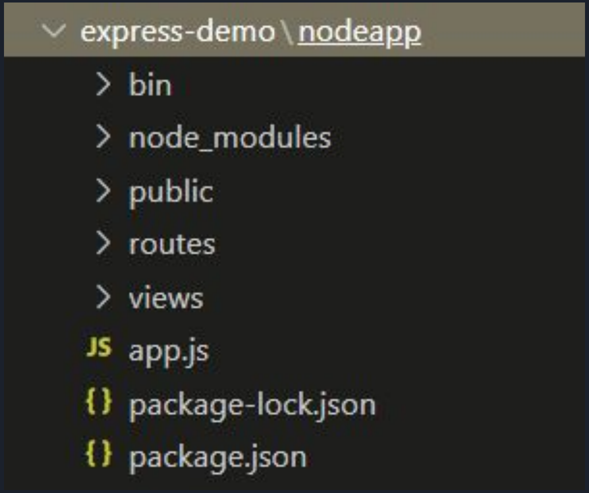
Go to the created root folder `myapp` by running the following command

```
PS D:\myproject>cd nodeapp
```

Install dependencies using the following command

```
PS D:\myproject>nodeapp >npm install
```

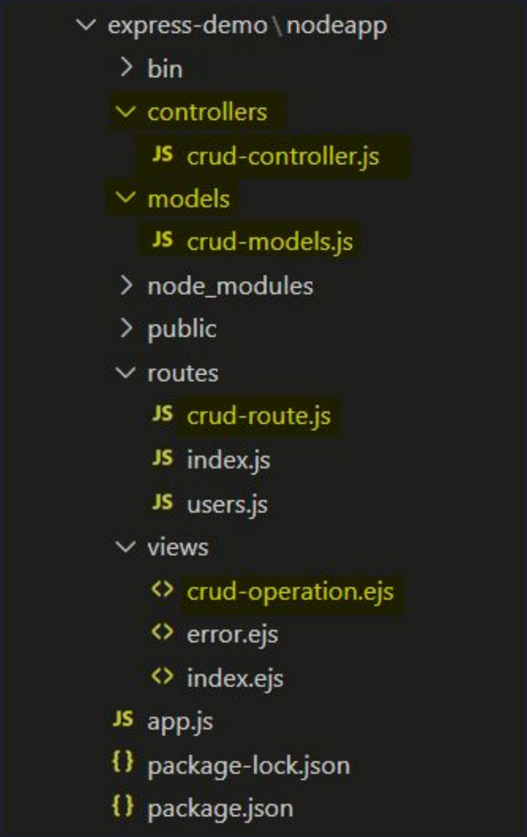
Basic folder structure of Express



```
∨ express-demo\nodeapp
    > bin
    > node_modules
    > public
    > routes
    > views
   JS app.js
   {} package-lock.json
   {} package.json
```

The default basic structure only has the view folders, so we have to create Controllers and models folder into it.

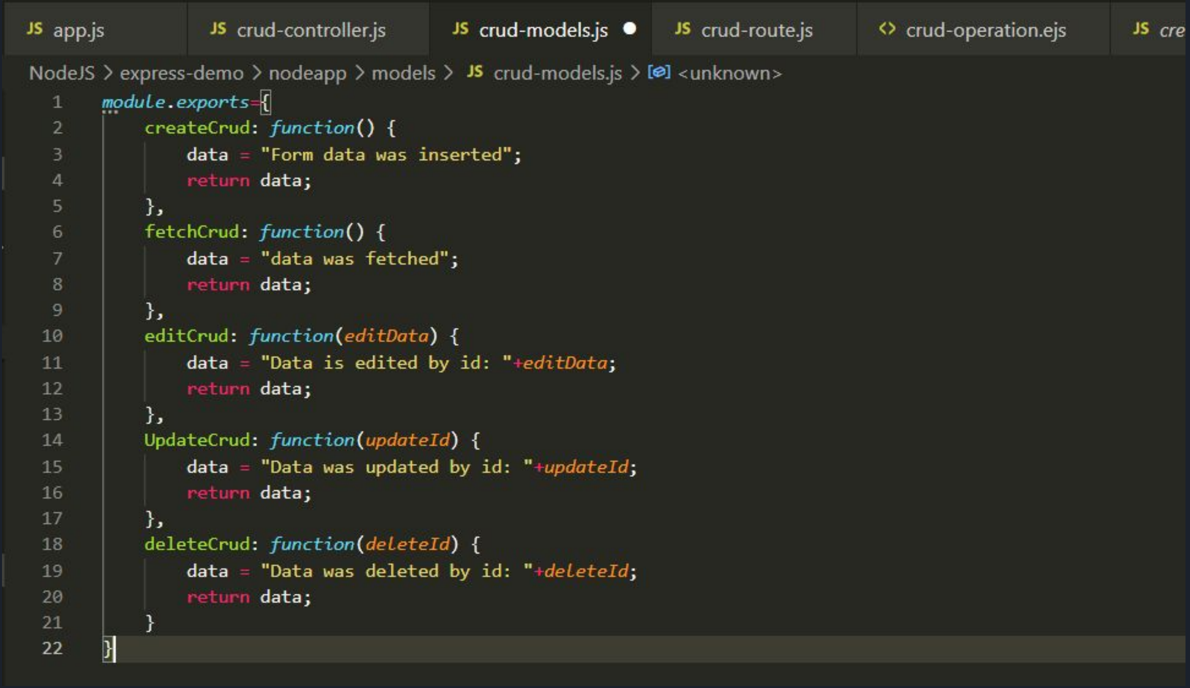
Folder Structure after adding controllers and models



```
express-demo\nodeapp
  bin
  controllers
    crud-controller.js
  models
    crud-models.js
  node_modules
  public
  routes
    crud-route.js
    index.js
    users.js
  views
    crud-operation.ejs
    error.ejs
    index.ejs
  app.js
  package-lock.json
  package.json
```

Express - Model

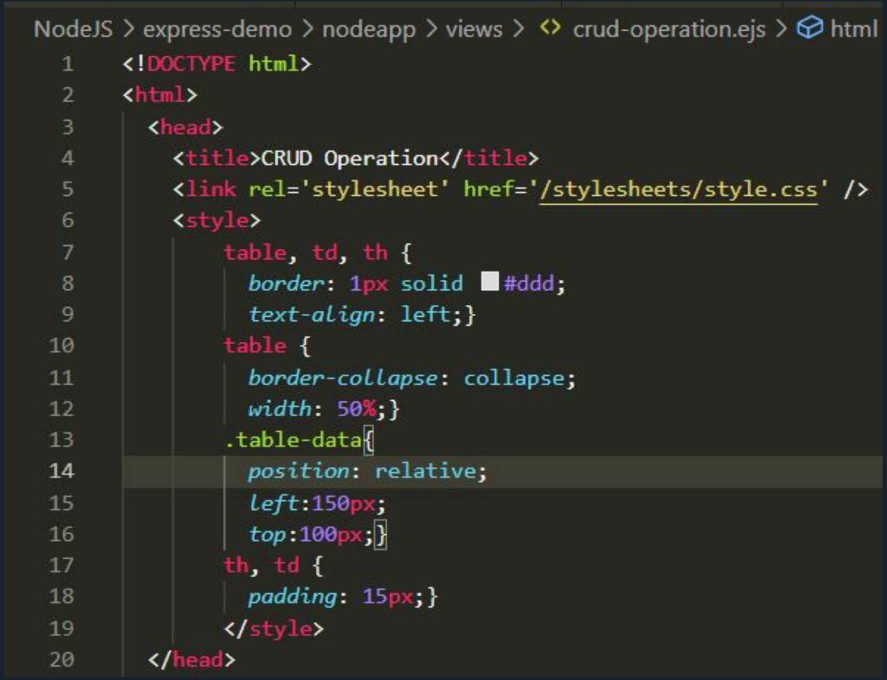
- You can write the functionality & logics related to the database like insert, fetch, update, delete queries.
- It also takes the query request from the controller & sends the response back to the controller.



```
module.exports={
    createCrud: function() {
        data = "Form data was inserted";
        return data;
    },
    fetchCrud: function() {
        data = "data was fetched";
        return data;
    },
    editCrud: function(editData) {
        data = "Data is edited by id: "+editData;
        return data;
    },
    UpdateCrud: function(updateId) {
        data = "Data was updated by id: "+updateId;
        return data;
    },
    deleteCrud: function(deleteId) {
        data = "Data was deleted by id: "+deleteId;
        return data;
    }
}
```

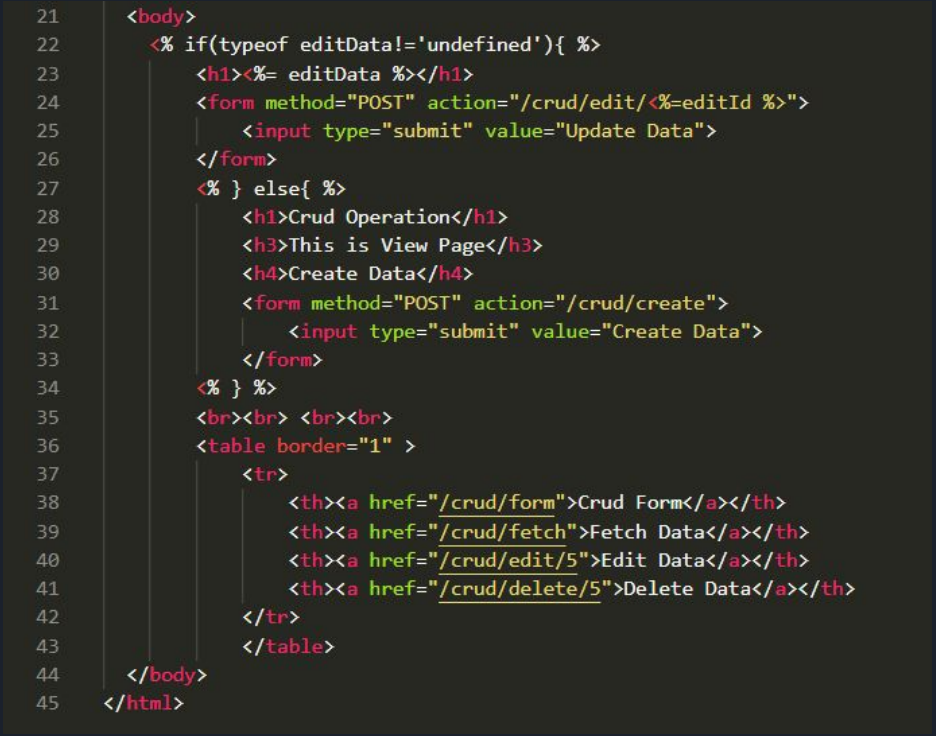
Express - View

- You can write HTML code for displaying a web page on the web browser.
- You can also send the data from the controller to view for displaying data dynamically.



NodeJS > express-demo > nodeapp > views > crud-operation.ejs > html

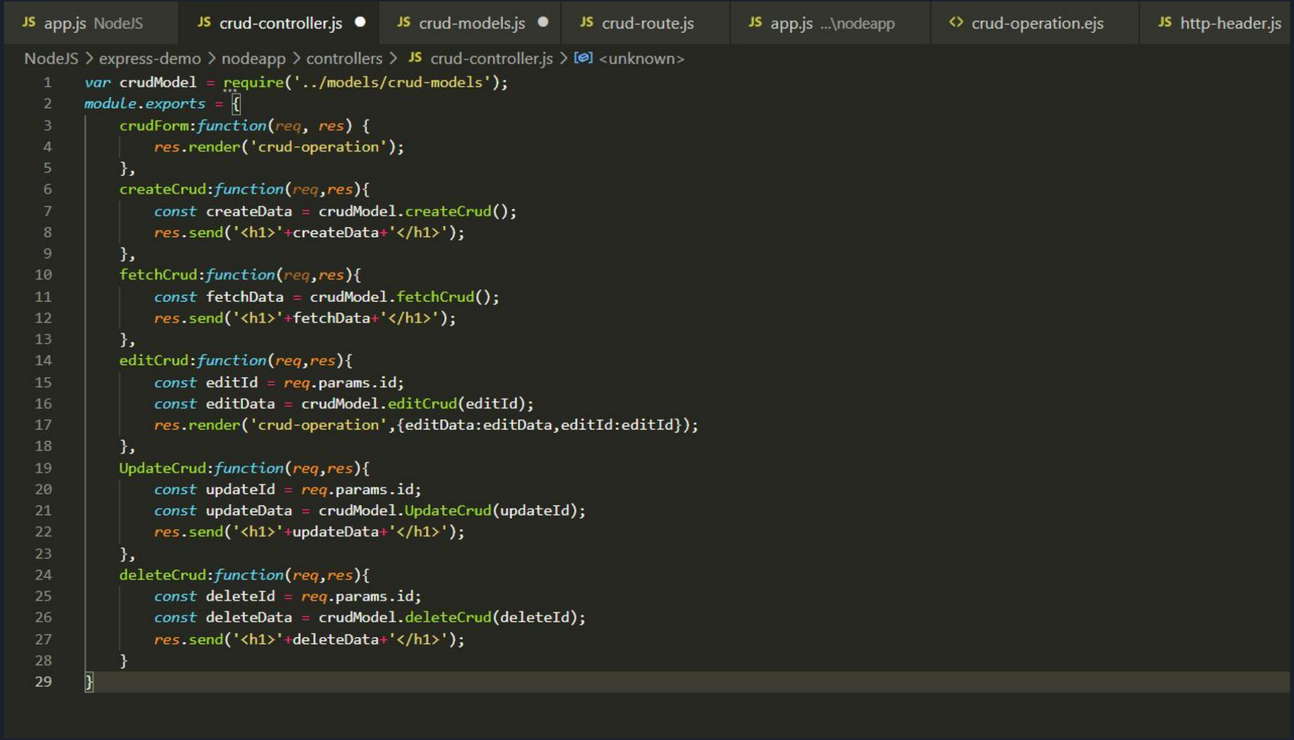
```
<!DOCTYPE html>
<html>
  <head>
    <title>CRUD Operation</title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
    <style>
        table, td, th {
          border: 1px solid #ddd;
          text-align: left;}
        table {
          border-collapse: collapse;
          width: 50%;}
        .table-data{
          position: relative;
          left:150px;
          top:100px;}
        th, td {
          padding: 15px;}
        </style>
  </head>
```



```
  <body>
    <% if(typeof editData!='undefined'){ %>
        <h1><%= editData %></h1>
        <form method="POST" action="/crud/edit/<%=editId %>">
            <input type="submit" value="Update Data">
        </form>
        <% } else{ %>
            <h1>Crud Operation</h1>
            <h3>This is View Page</h3>
            <h4>Create Data</h4>
            <form method="POST" action="/crud/create">
                <input type="submit" value="Create Data">
            </form>
        <% } %>
        <br><br> <br><br>
        <table border="1" >
            <tr>
                <th><a href="/crud/form">Crud Form</a></th>
                <th><a href="/crud/fetch">Fetch Data</a></th>
                <th><a href="/crud/edit/5">Edit Data</a></th>
                <th><a href="/crud/delete/5">Delete Data</a></th>
            </tr>
            </table>
  </body>
</html>
```

Express - Controller

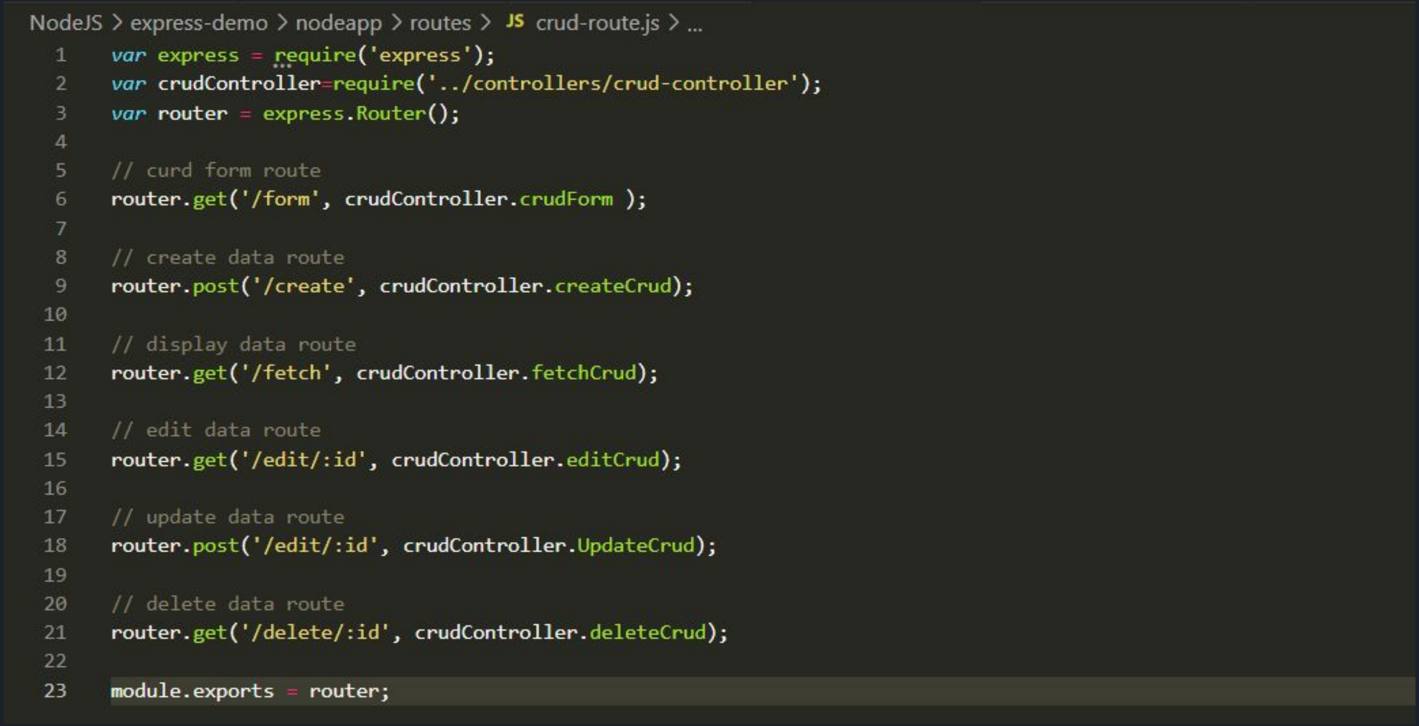
- You can write the functionality & logic to develop dynamic web applications.
- It can also take the data request from the views & send it to the model and send the response back to the views.



```
var crudModel = require('../models/crud-models');
module.exports = {
    crudForm:function(req, res) {
        res.render('crud-operation');
    },
    createCrud:function(req,res){
        const createData = crudModel.createCrud();
        res.send('<h1>'+createData+'</h1>');
    },
    fetchCrud:function(req,res){
        const fetchData = crudModel.fetchCrud();
        res.send('<h1>'+fetchData+'</h1>');
    },
    editCrud:function(req,res){
        const editId = req.params.id;
        const editData = crudModel.editCrud(editId);
        res.render('crud-operation',{editData:editData,editId:editId});
    },
    UpdateCrud:function(req,res){
        const updateId = req.params.id;
        const updateData = crudModel.UpdateCrud(updateId);
        res.send('<h1>'+updateData+'</h1>');
    },
    deleteCrud:function(req,res){
        const deleteId = req.params.id;
        const deleteData = crudModel.deleteCrud(deleteId);
        res.send('<h1>'+deleteData+'</h1>');
    }
}
```

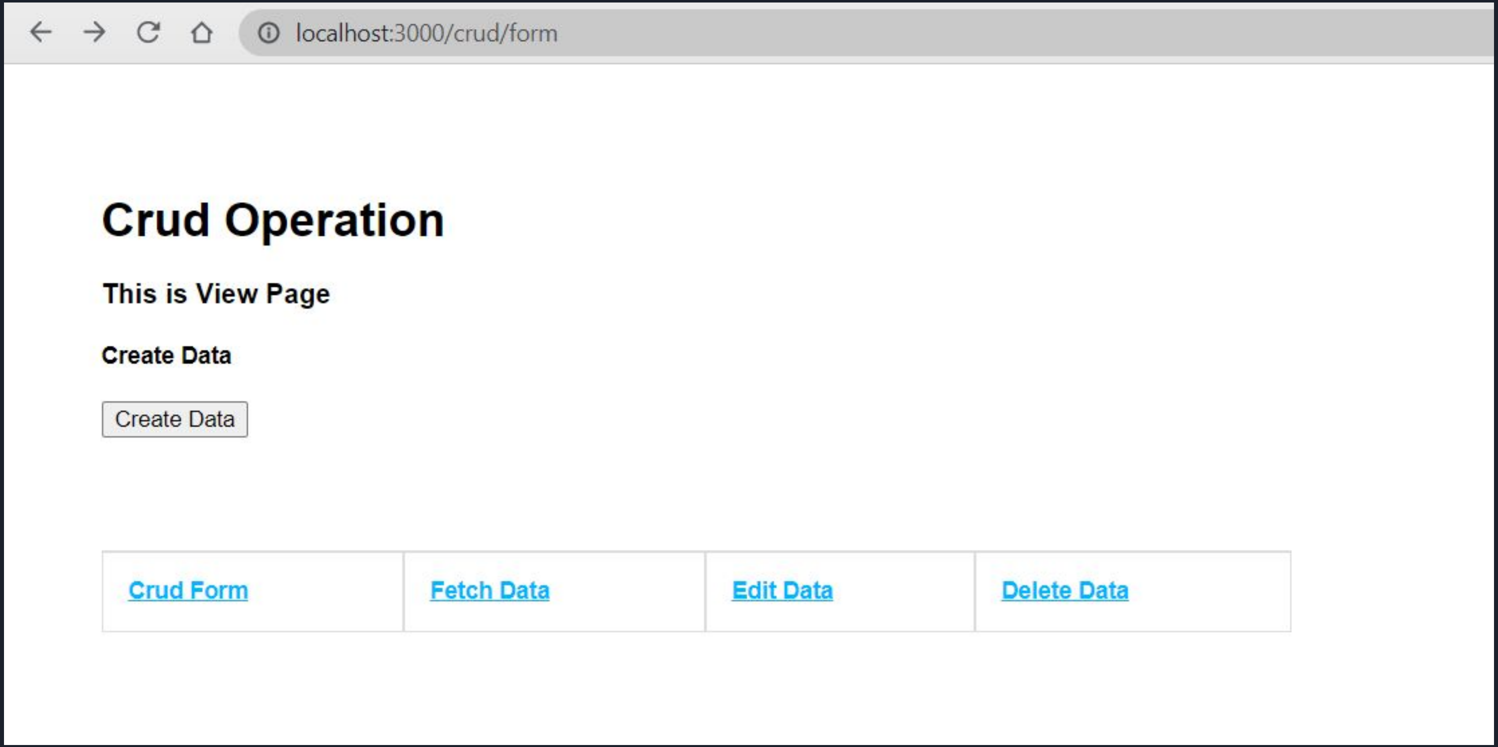
Express – Route

In the route folder, you can create a custom route/link to execute the dynamic web pages.



NodeJS > express-demo > nodeapp > routes > JS crud-route.js > ...

```
var express = require('express');
var crudController=require('../controllers/crud-controller');
var router = express.Router();

// curd form route
router.get('/form', crudController.crudForm );

// create data route
router.post('/create', crudController.createCrud);

// display data route
router.get('/fetch', crudController.fetchCrud);

// edit data route
router.get('/edit/:id', crudController.editCrud);

// update data route
router.post('/edit/:id', crudController.UpdateCrud);

// delete data route
router.get('/delete/:id', crudController.deleteCrud);

module.exports = router;
```



localhost:3000/crud/form

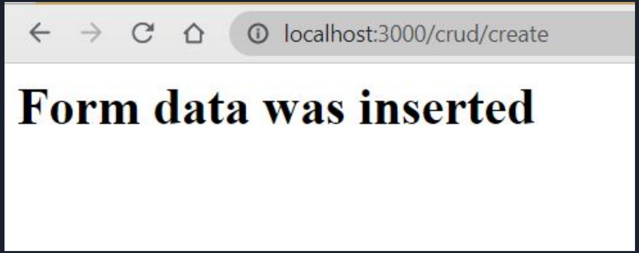
Crud Operation

This is View Page

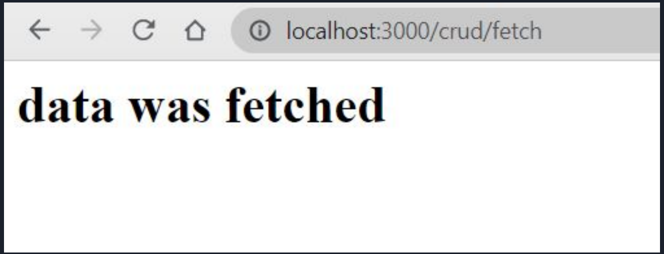
Create Data

Create Data

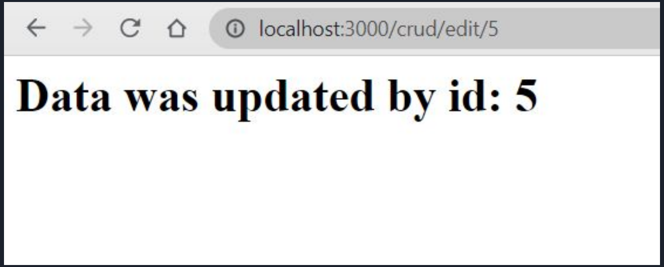
Crud Form	Fetch Data	Edit Data	Delete Data


localhost:3000/crud/create

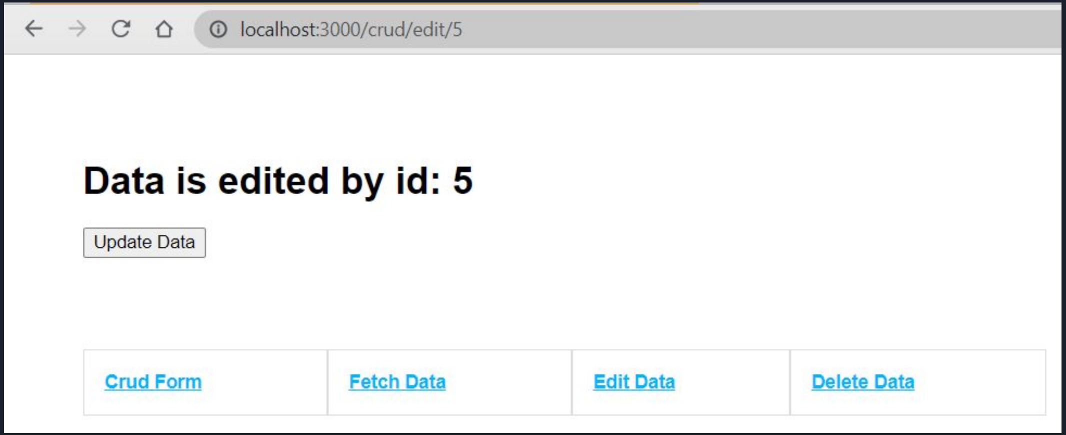
Form data was inserted


localhost:3000/crud/fetch

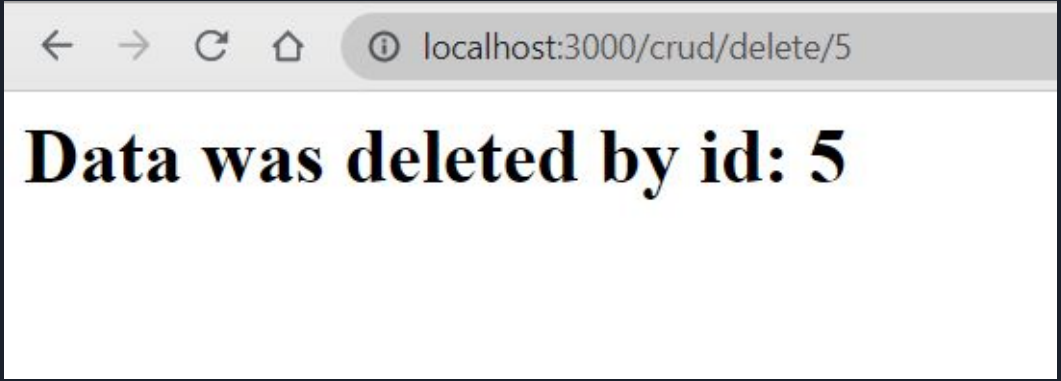
data was fetched


localhost:3000/crud/edit/5

Data was updated by id: 5


localhost:3000/crud/edit/5

Data is edited by id: 5


localhost:3000/crud/delete/5

Data was deleted by id: 5

THANK YOU.