Blocking & Non Blocking :-

Blocking:- is when the execution of additional JS in the Node.js process must wait until a non-JS operation completes.

→ I/o refers primarily to interaction with the system's disk & methods supported by libuv.

Blocking I/o

```
var fs = require("fs");   → imported
// synchronous read.
var data = fs.readFileSync('input.txt')
```

more method will be blocked till the read method is not executed.

Non-Blocking I/o

```
var fs = require("fs")
// asynchronous read.
fs.readFile('text.txt',
    function (err, data)
{
});
```

more method will execute asynchronously

Non-blocking:- ~~it refers~~ to the one executed asynchronously.

~~Blocking.js~~ ~~synchronously~~ means that the program may not exate line by line.

```
var fs = require('fs');
fs.readFile('test.txt', function(err, data){
                ↑
        // Asynchrous funiction
  if (err){
      console.log(err);
  }
  set Timeout(() => {   // It execute one-time after delay
      console.log('Display'); millisecond, specify here
  }, 2000);
});
```

Node modules;-

1. Npm
2. Globals
3. file-system
4. call back
5. event
6. HTTP

Npm;- Node package Manages

→ provides online repositories for node.js packages/modules

→ provide command line utility to ~~un~~install Node.js packages

npm install → install all the module as specified in package.json

npm install <module Name> → install module using npm

npm install <module Name> -g → install dependency globally

2. **Global objects**;- the objects are avaiable in all modules

__dirname;- specify the name of the directory that currently contains the code.

__filename;- specifies the name of the file that currently contains the code

> A timer in Node.js is an internal construct that calls a given function after a certain period of time.

1. setTimeout(callback, delay[,...args])
2. setInterval(callback, delay[,...args])
3. setImmediate(callback, [,....args])

③ file system:-

file i/o is provided by simple wrappers around standard POSIX function's

portable os

var fs = require('fs');

fs Method's

Synchronous forms

```
var fs = require('fs');
// sync
var data = fs.readFileSync('input.txt');
```

Asynchrony form

```
var fs = require('fs');
// ASY
fs.readFile('text.txt',
    function(err, data)
    {
    }
);
```

callback as Last Arg

Method's in file system module:-

opening a file

fs.open(path, flags[, mode], callback) → open file (asynchronously)

fs.openSync(path, flags[, mode]) → open file synchronously

fs.close(fd, callback) → closing file

Reading from a file

read(fd, buffer, offset, length, position, callback)
↳ Read content of a file into Buffer

readFile(file [, options], callback)
↳ Reads file Asynchrony

readFileSync(file [, options])
↳ Read, file synchron

writeFile(file, data[, options], callback)
↳ openfile asyn.
writeFileSync(file, data[, options]) → open file synchronously
→ writing in a file

(4)

Callback:-

callback is an asynchronous equivalent for a function & is called at the completion of each task.

callback:- will execute after file read is complete.

```
var fs = require("fs");
fs.readFile('text.txt', function(err, data){ });
```

(5) Events:-

→ Node.js as follows event-driven architetuge.
→ certain objects (emitters) periodically emit events which further invoke's the listners.
→ Node.js poovide concurrency by using the concept of events Eg call backs
→ all objects that emit event's are instance of the EventEmitter class.

```
var fs = require('fs');
var event = require('event');  ---> import events module
const myEmitter = new event.EventEmitter();
```
↳ creating object of event Emitter.

```
fs.readFile('test1.txt', (err, data) => {
    console.log(data.toString());
    myEmitter.emit('readFile');  ---> Emitting event
});
```

```
myEmitter.on('readFile', () => {
    console.log('In Read Event occured');
});
```
→ Registering Listeners & defining event handlers

HTTP: → To use the http server & client one must require('http')
→ The http interfaces in Node.js are designed to support many features of the protocol.

```
var http = require('http');
var fs = require('fs');
var url = require('url');
```
→ import Required modules

```
http.createServer(function (request, response) {
    var pathname = url.parse(request.url).pathname;
    console.log("Request for " + Pathname + "received");
    fs.readFile(pathname.substr(1), function (err, data) {
        if (err) {
            console.log(err);
            response.writeHead(404, {'content-Type': 'text/html'});
```
→ creating server
→ parse the fetched URL to get pathname
→ Req file to be read from file system (index.html)
↳ Creating header with content type as text or HTML

```
}else{
    response.writeHead(200, {'Content-Type': 'text/html'});
    response.write(data.toString());
}
response.end();
});
}).listen(3000);
console.log('Server running at localhost:3000');
```

↳ Generating Response.

↳ listening to port: 3000

* Node.js

V8 Engine:- V8 is Google's open source high-performance JS & webassembly engine, written in C++.
It is used in Chrome & in Node.JS
V8 can start standlone, or can be embeded into any C++ appl?

what is Node.JS?-

→ powerful JS framework.
→ developed on chrome's V8 JS-engine.
→ compiles JS directly into the native machine code.
→ used for creating server-side web appl? ~~native mach~~
→ Right pick for the data-intensive real time appl?

features;

1. open source
2. simple & fast
3. Asynchronous
4. High Scalability
5. single Thread
6. No-Buffering
7. cross platform → ~~of~~ any platform's

Npm :- Node package module.
It is a package manager for Node.js packages modules which has been added as defaul in the node installation

Node modules :- search

① core module :- http, url, path, fs - .
local " →
3rd party " → it's aviable in there modules developed by the other developers & free to use. few of the best external modules are express, react . . .

JSON file :- the package.Json file in Node.js is the heart of the entire appl^n. It is basically the manifest file that contains the meladata of the project

Node Js Datatypes :-

Primitive D-T
1 String
2 Number
3 Boolean
4 Null
5 undefined

Non-primitive DT
1 object
2 Data
3 Array

function's:- function in Node.js is a block of code that has a name & is written to achieve a particular task. you need to use the keyword function to create it. function is generally a two step process first is defining the function & 2nd is invoking it.

file system:-

To access the physical file system. Node.js makes use of the fs module which basically takes care of all synchronous & asynchronous file I/o operations.

syntax: var fs = require('fs');

file system module's:- Read "
create "
update "
delete "
Rename "

// Importing module's
var myEvent = require('events');

Binding event to an event listener

var myEventEmitter = new myEvents.EventEmitter();

Event:-
→ Emit it method
- on "

→ myEventEmitter.on('eventName', eventHandler);

Binding event handler to an event

// fire an even
→ myEventEmitter.emit('eventName');

filling an event

HTTP module:-

Express.Js:- this is framework built-on-top of Node.Js that facilitate the management of the flow of data blw server & routes in the server-side appl^n.

⇒ It is a lightweight & flexible framework that provides a wide range of features required for the web as well as mobile appl^n development

→ fastw appl^n

→ Provides two templating engines

→ Helps in building appl^n of single-page & hybrid types

→ makes integration with db easy.

→ simplifies configuration & customization of the appl^n

→ Defines defines an error handling middleware.

what is Npm:-

It is responsible for managing all the Node.js packages & module's present in an appl?

features of default package manage of Node.js

→ it is complete open-source written in JS
→ It's is would largest registry → it manges all the packages & module's
→ free-& open source

main function of Npm:-

① provides online-repository for package's / module's for Node.JS which you can easily search online on their official site
② It is also provides CLI which helps the developers in locally interacting with their system's

used for Npm:-

Helps in incorporating the pre-built package's into our project.
⇒ Asists in dealing various standlone tool's which can be used right way
using npx. you can even run packages without having to deal it
developers often use Npm to share their code with other Npm users

Npm packages:-

npm express :- It makes very easy to develop an appl? which can be used to handle multiple type of request
e.g. get, put, del - -

npm install express

* npm body-parser;- it extracts the entire body portion of an incoming req stream
⇒ it is middleware. we have to install seperatly
npm install body-parser
→ it passes the JSON buffer string & URL encoded data

③ npm nodemon;- it will utility that monitor for any changes in your source. it will automatically restar your servy

npm install -g nodemon

④ ~~npm babel-core;-~~

⑤ npm lodash;- here we can create composite functions manipulate object's & some manipulation's

⑥ npm react;- used to create React-app's

⑦ npm request;- it is design for simplest way possib to make HTTP call

⑧ npm async;- it is utility module. it provide's straightforward powerful function's

Routing Methods:-

Get → send data to client

post → receive data from the client side & stores in db

put → update is used updating content

delete → delete the data.

---

① npm install express -generator -g ↵

if get error

sudo npm install " "

→ new floder ↵

② — express ↵

③ npm install

④ npm start