Node.js

Node.js

JavaScript on the server

Asynchronous event-driven JavaScript runtime

Lightweight

Designed to build scalable runtime applications

Designed with streaming and low latency in mind

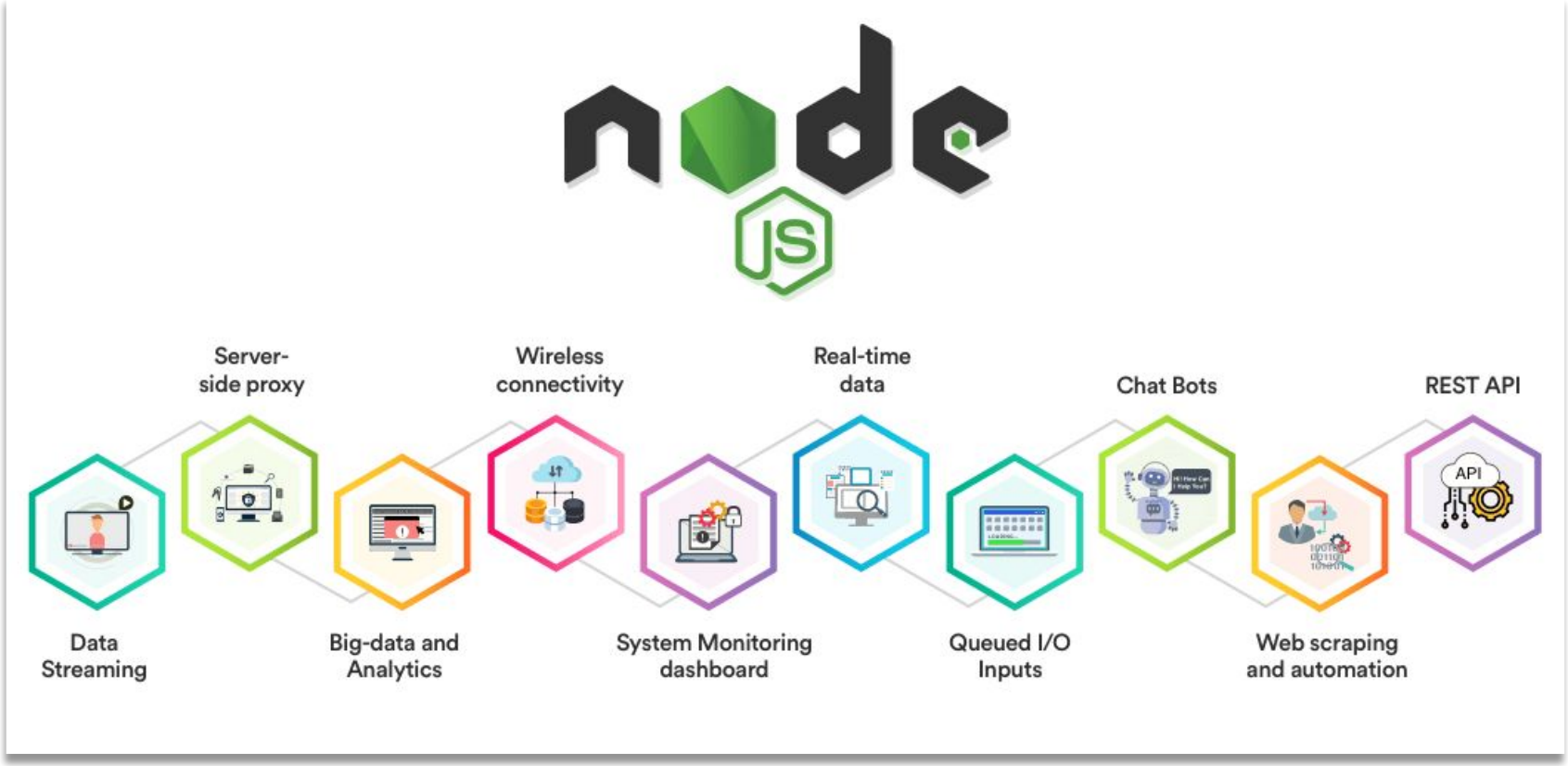
Makes Node.js as a foundation of web library or framework

Built on V8 (Google's opensource high-performance JavaScript)

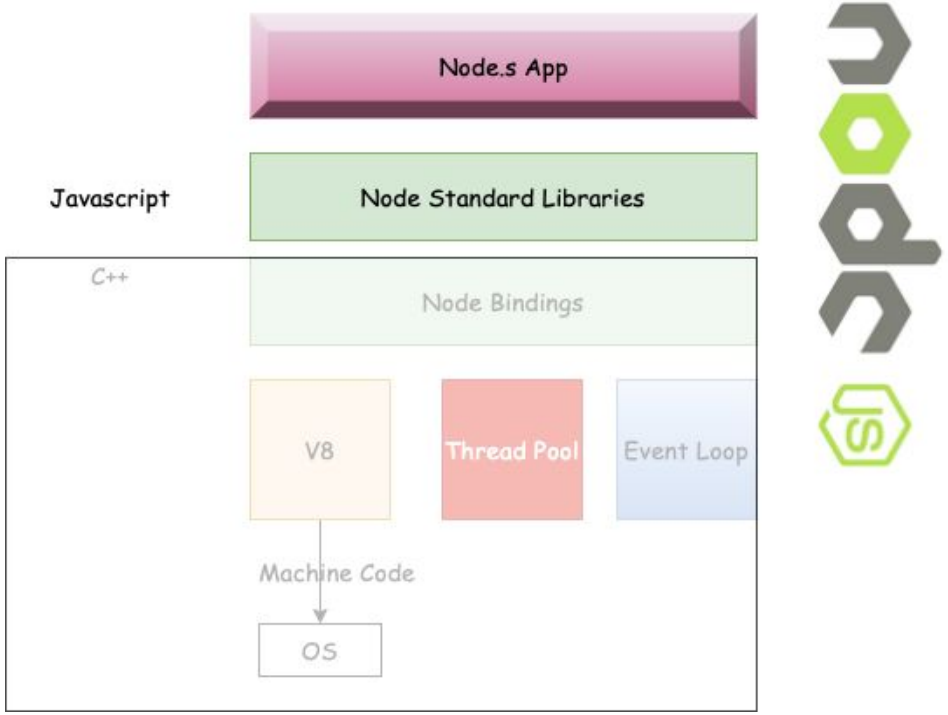
Easy to learn

Massive library support

Use-cases of Node.js

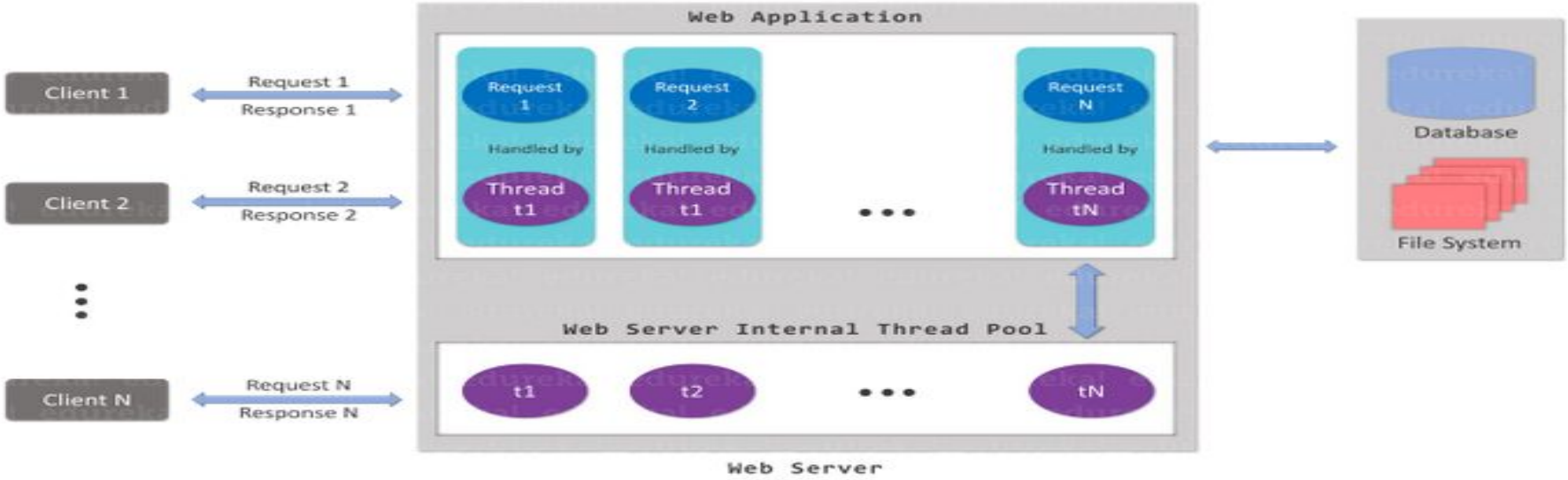


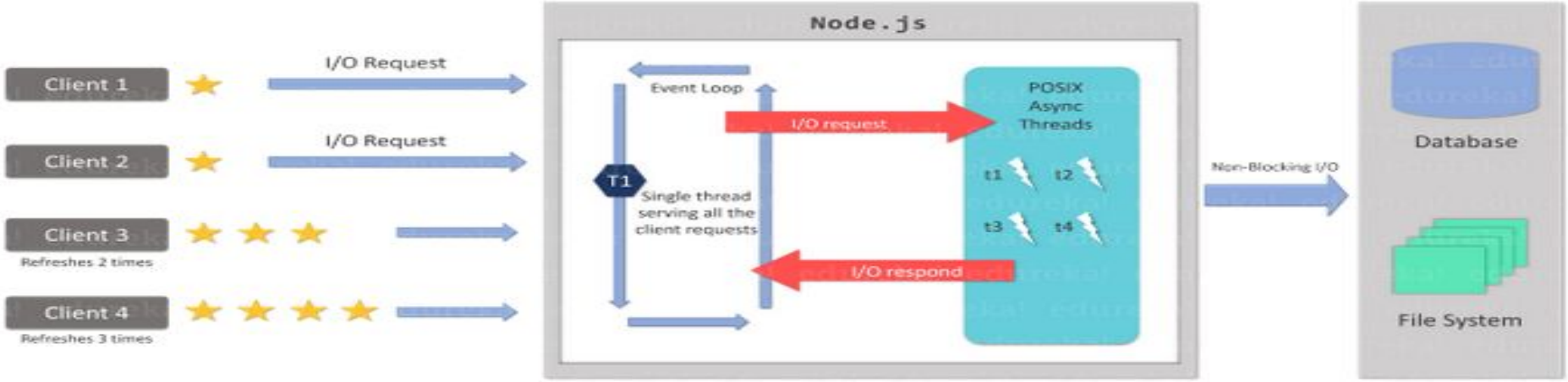
Node.js Internals



Difference between Node.js & JS

JavaScript	Node.js
Used for writing scripts on the website.	JavaScript runtime environment
JavaScript can only be run in the browsers	JavaScript on the server side (outside browser)
capable to work with HTML and play with the DOM.	Does not have the capability to add HTML tags
Can run in any browser engine as like JS core in safari, Firefox..	Can only run in V8 engine.
JavaScript is used in frontend development.	Nodejs is used in server-side development.
Some of the JavaScript frameworks are RamdaJS, TypeJS, etc.	Some of the Nodejs modules are Lodash, express etc. These modules are to be imported from npm.


Web Application
Client 1
Request 1
Response 1
Client 2
Request 2
Response 2
Client N
Request N
Response N
Request 1
Handled by
Thread t1
Request 2
Handled by
Thread t1
Request N
Handled by
Thread tN
Web Server Internal Thread Pool
t1
t2
tN
Web Server
Database
File System


Node.js
Client 1
I/O Request
Client 2
I/O Request
Client 3
Refreshes 2 times
Client 4
Refreshes 3 times
Event Loop
I/O request
T1
Single thread serving all the client requests
I/O respond
POSIX Async Threads
t1
t2
t3
t4
Non-Blocking I/O
Database
File System

Install Node.js

Website

- https://nodejs.org/en/

Check

- node –version

Package

- npm install <package-name>

Initiate

- node init

Node.js Examples

```
console.log('Welcome to FSD-2 course');
```

Node.js writing a function

```
function Hello() {
    console.log('Welcome to FSD-2 course');
}
Hello();
```

Node.js writing a function

```
function Hello(subject) {
    console.log('Welcome to ' + subject +' course');
}
Hello("FSD-2");
```

Node.js Modules

- Node programs can be organized as modules
- A module is a file that's exports a scope – file management
  - Contains public functions
  - Contains shared objects
- Modules are imported through the *require* function

Three types of modules

- Core Modules – Built-in modules part of the platform
- Local Modules – Application based modules
- Third-party Modules – Third party modules

Core Modules

Core Modules	Description
http	creates an HTTP server in Node.js.
assert	set of assertion functions useful for testing.
fs	used to handle file system.
path	includes methods to deal with file paths.
process	provides information and control about the current Node.js process.
os	provides information about the operating system.
querystring	utility used for parsing and formatting URL query strings.
url	module provides utilities for URL resolution and parsing.

Local Modules

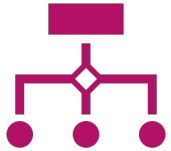
```
var logg = require('./3-1');

console.log(logg);

logg.log1('Welcome to FSD-2');
```

```
var url ='http://iiits.in/';

function log(message){
    //send http request
    console.log(message);
}

module.exports.log1 = log;
//exports.log1 = log;
module.exports.Linkurl =url;
```

Third-party Modules

Third-party modules are modules that are available online using the Node Package Manager(NPM)

These modules can be installed in the project folder or globally



Sample third-party modules

mongoose, express, angular, and react.

Node Package Manager

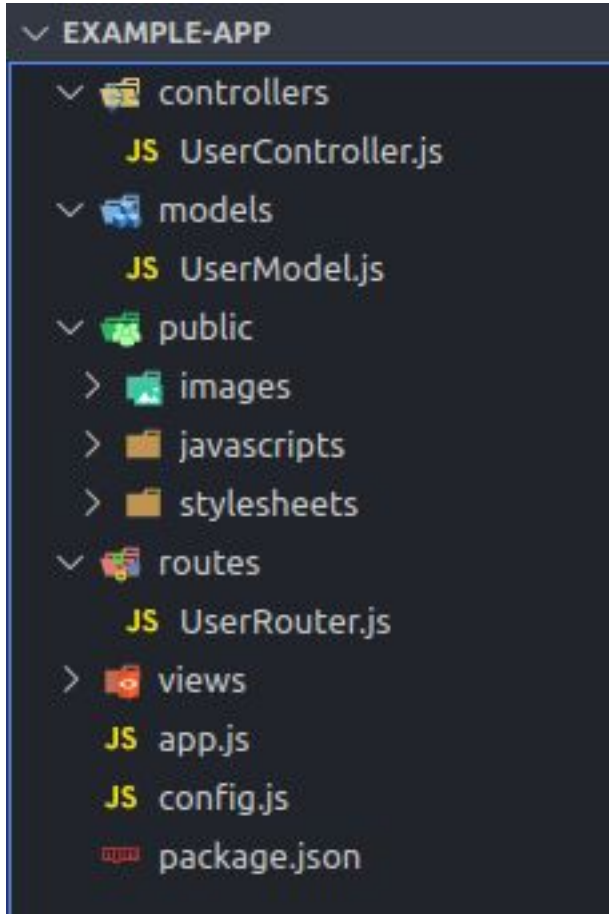
Online repository for open-source Node.js packages

Node Package Manager (NPM) is a command line tool that installs, updates or uninstalls Node.js packages in your application

The node community around the world creates useful modules and publishes them as packages in this repository

```
npm install npm –g
npm install <package name>
npm install express
```

Node.js application structure



app.js:- This file starts your web server. All your set up logic should be in this file.

Controllers:- This folder contains all the business logic of your application.

Models:- All the database models should go into the models folder.

Public:- All the public files such as images, javascript files, CSS files should go into this folder.

Routes:- All your routing-related logic should go into this folder

Views:- So this folder contains all your views i.e. HTML/ejs files. Drop this folder if you are building rest API's.

config.js:- This file should contain all your configuration e.g. PORT number, secrets, keys etc.

package.json

- Manifest file for your project
- Lists all the installed packages

```
{
  "name": "package.json-mastery",
  "version": "1.0.0",
  "description": "Mastery of the package.json file",
  "main": "index.js",
  "scripts": {
    "start": "node index",
    "dev": "nodemon index",
    "test": "jest"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/Easybuoy/package.json-mastery.git"
  },
  "keywords": [
     "javascript",
     "npm”
   ],
  "author": ”Author name",
  "homepage": "https://github.com/Easybuoy/package.json-mastery#readme",
  "engines": {
     "npm": "6.10.0",
     "node": "10.14.1"
   },
   "dependencies": {
     "bcryptjs": "^2.4.3",
     "cors": "^2.8.5",
     "dotenv": "^6.1.0",
     "express": "^4.16.4"
   },
   "devDependencies": {
     "eslint": "^4.19.1",
     "mocha": "^6.2.0",
     "nodemon": "^1.19.1"
   },
   "nyc": {
     "exclude": [
       "server/app.js",
       "server/config/",
       "server/build"
     ]
   }
}
```

package-lock.json

- File listing the full *dependency tree* of your project.


```
{
"requires": true,
"lockfileVersion": 1,
"dependencies": {
"abbrev": {
"version": "1.1.1",
"resolved": "https://registry.npmjs.org/abbrev/-/abbrev-1.1.1.tgz",
"integrity":
"sha512-nne9/IiQ/hzIhY6pdDnbBtz7DjPTKrY00P/zvPSm5pOFkl6xuGrGnXn/VtTNNfNtAfZ9/1RtehkszU9qcTii0Q==",
"dev": true
},
"accepts": {
"version": "1.3.5",
"resolved": "https://registry.npmjs.org/accepts/-/accepts-1.3.5.tgz",
"integrity": "sha1-63d99gEXI6OxTopywIBcjoZ0a9I=",
"dev": true,
"requires": {
"mime-types": "~2.1.18",
"negotiator": "0.6.1"
}
}
}
}
```

Event Emitters

- **Event-driven programming** is a programming paradigm in which the flow of the program is determined by events. An event-driven program performs actions in response to events. When an event occurs it triggers a callback function.

Observer pattern

Event Emitter Methods

EventEmitter Methods	Description
emitter.addListener(event, listener)	Adds a listener to the end of the listeners array for the specified event. No checks are made to see if the listener has already been added.
emitter.on(event, listener)	Adds a listener to the end of the listeners array for the specified event. No checks are made to see if the listener has already been added. It can also be called as an alias of emitter.addListener()
emitter.once(event, listener)	Adds a one time listener for the event. This listener is invoked only the next time the event is fired, after which it is removed.
emitter.removeListener(event, listener)	Removes a listener from the listener array for the specified event. Caution: changes array indices in the listener array behind the listener.
emitter.removeAllListeners([event])	Removes all listeners, or those of the specified event.
emitter.setMaxListeners(n)	By default EventEmitters will print a warning if more than 10 listeners are added for a particular event.
emitter.getMaxListeners()	Returns the current maximum listener value for the emitter which is either set by emitter.setMaxListeners(n) or defaults to EventEmitter.defaultMaxListeners.
emitter.listeners(event)	Returns a copy of the array of listeners for the specified event.
emitter.emit(event[, arg1][, arg2][, ...])	Raise the specified events with the supplied arguments.
emitter.listenerCount(type)	Returns the number of listeners listening to the type of event.

Local Modules(ES6)

```
import {fsd} from "./10.mjs";
import {wad} from "./10.mjs";
import * as fsdc from "./10.mjs";
fsdc.fsd();
fsdc.wad();
```

```
export function fsd(){
    console.log("FSD-1 is a part of FSD")
}

export function wad(){
    console.log("WAD is not a part of FSD")
}
```

Core Modules: **path**

- filename
- dirname

path.parse() returns elements of the path.

Properties:

- dir
- root
- base
- name
- ext

Core Modules: **os**

- os.freemem()
- os.getPriority([pid])
- os.homedir()
- os.hostname()
- os.loadavg()
- os.networkInterfaces()
- os.platform()
- os.release()
- os.setPriority([pid,]priority)
- os.tmpdir()
- os.totalmem()
- os.type()
- os.uptime()

……..

console.count()

Maintains an **internal counter** and outputs to **stdout** the number of times console.count()

Third party Module: **chalk**

chalk: color your font (https://www.npmjs.com/package/chalk)

chalk.blue()

chalk.red()

……..

Third party Module: **progress**

ProgressBar: Show the progress of a task (https://www.npmjs.com/package/progress)

Readline function

Define Readline function

```
const readline = require('readline').createInterface({
    input: process.stdin,
    output: process.stdout
})
```

Taking i/p and print

```
readline.question('Course Name ', course =>{
        console.log('Course name is ' +course)
        readline.close()
})
```

Interval

Set Interval

Clear Interval

```
const interval = setInterval(()=>{
    if(2==3){

        clearInterval(interval)
        return
    }
    console.log('FSD1');
}, 10)
```

Time Out

```
setTimeout(function(){
    console.log('FSD');
}, 1000)

console.log('Welocme to');
```

What will be the output?

Event Emitter

```
const evnte = require('events');
const emt = new evnte();

// a listener
emt.on('msg',()=>{
    console.log('Welcome to FSD-1')
})

//trigger the event
emt.emit('msg');
```

Event Emitter

```
const evnte = require('events');
const emt = new evnte();

emt.on('FSD Project Submission', ()=>{
    console.log('Please submit on time');
    setTimeout(()=>{
        console.log('Last day for submission');
    }, 8000)
    setTimeout(()=>{
        console.log('Its a gentle reminder');
    }, 3000)
})
console.log('Submission starts')
console.log('Submission starts from tomorrow')
emt.emit('FSD Project Submission');
```

What will be the output?

FS Modules

```
const fs = require('fs');
//
try {
    const fd = fs.openSync('file', 'r')
} catch(err){
    console.error(err)
}
```

```
fs.readFile('file2', (err,data)=>{
    if(err) throw err;
    console.log(data.toString())
})
```

```
const content = 'FSD2 is part FSD track'
fs.writeFile('file2', content, (err)=>{
    if(err) throw err;
    console.log('Done Successfully')
})
```

```
fs.appendFileSync('file1', 'FSD3 is also part of FSD Track', (err)=>{
    if(err) throw err;
    console.log('Done Successfully')
})
```

http Modules

```
const http = require('http');
const server = http.createServer();

server.listen(3001);
console.log('server is working')
```

Express

```
app.get('/', (req, res) => {
  res.send('Hello World!')
})
```

EJS(Embedded JavaScript templates)

```
<%- include('head'); %>
<%- include('navbar'); %>
<p>FSD1 is part of FSD
```

EJS(Embedded JavaScript templates)

```
<%- include('head'); %>
<%- include('navbar'); %>
<p>FSD1 is part of FSD
```

STATIC(Middleware)

```
app.use(express.static(path.join(__dirname, 'public')));
```

Body Parser(Middleware)

Define

```
const bparser = require('body-parser')
```

```
app.use(bparser.urlencoded({extended:false}));
```

Rendering

```
res.render('abt', {fname: req.body.firstname, lname: req.body.lastname});
```

In Memory Database

```
const sqlite3 = require('sqlite3')
```

sqlite3

Database connection

```
const db_name = path.join(__dirname, "data", "fsdapp.db");
const db = new sqlite3.Database(db_name, err =>{
    if(err){
        return console.log(err.message);
    }
    console.log("FSD Database Connected")
});
```

sqlite3

Create Table

```
const ousers = `CREATE TABLE users(
    uid INTEGER PRIMARY KEY AUTOINCREMENT,
    firstname VARCHAR(100) NOT NULL,
    lastname VARCHAR(100) NOT NULL

);`;
```

```
db.run(ousers, err=>{
    if(err){
        return console.log(err.message)
    }
    console.log("FSD User table created successfully")
})
```

sqlite3

Insert Data

```
const sinsert = `INSERT INTO users (uid, firstname, lastname) VALUES
(1, 'Himangshu', 'Sarma'),
(2, 'ABC', 'DEF')`;
db.run(sinsert, err =>{
    if(err){
        return console.log(err.message)
    }
    console.log("FSD user details entered")
})
```

sqlite3

Display In a page

```
app.get("/FSD", (req, res) =>{
    const sql = "SELECT * FROM users ORDER by uid";
    db.all(sql, (err, rows)=>{
        if (err){
            return console.log(err.message);

        }
        res.render("fdata", {model: rows});
        res.render()
    })

})
```