Introduction To Node js

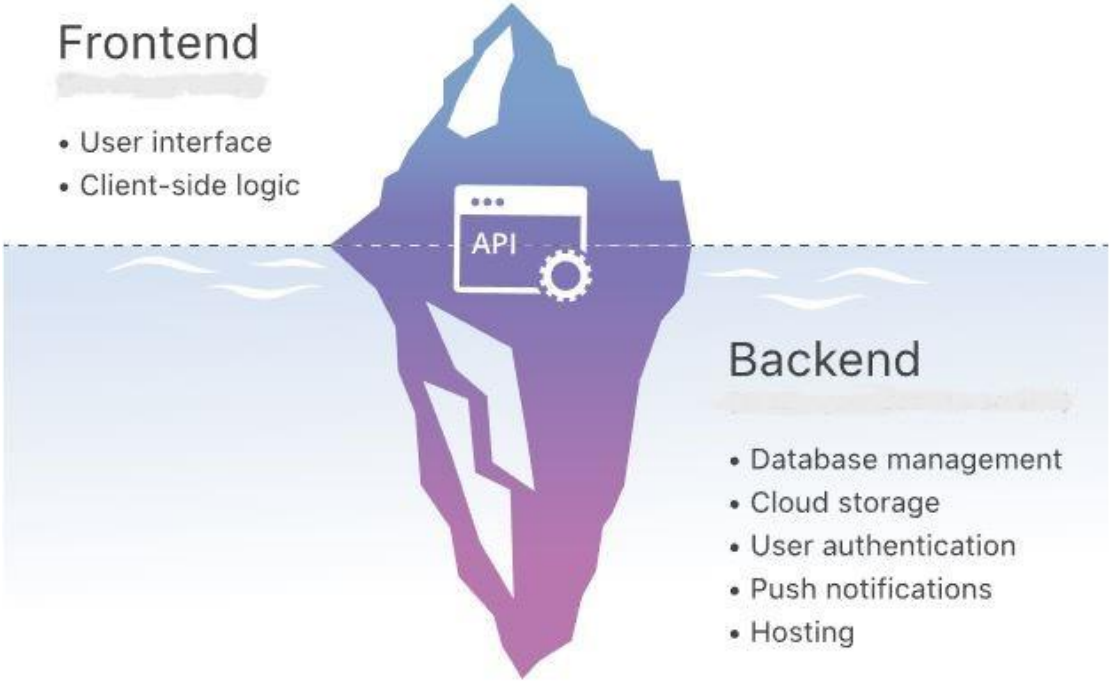
Computer Coding Club MNNIT

CONTENTS

What is Backend?

It is the part of our web application that is hidden from the users or clients.

It is also popularly referred to as the **“server-side”**.



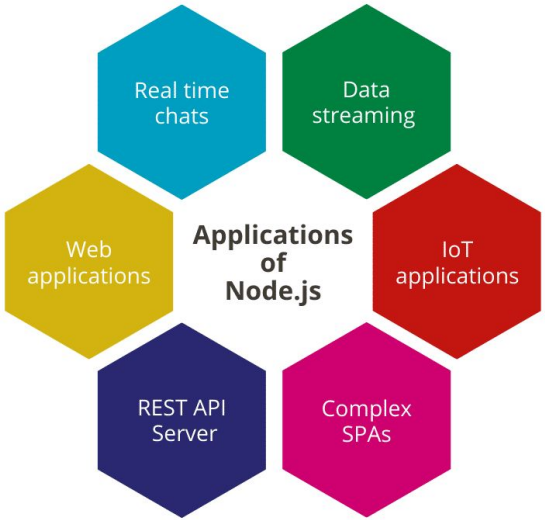
Why use backend?

- Our computers can hold limited information, so it acts like a digital store.
- Manages and stores user information and details securely.
- Implementation and integration of delicate services such as banking transactions etc.

What is Node JS?

Node JS is an open source runtime environment for creating server-side applications in JS.

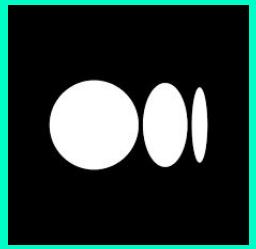
Node JS = Runtime Environment + JS Libraries



Some features of Node JS are :

- ❑ Asynchronous
- ❑ Very Fast
- ❑ Single Threaded and Highly Scalable
- ❑ Multiple Platform Compatible

NODE JS IN ACTION



Node JS setup

Node JS runtime environment can be installed from its official website.

- Official Website
- Installation

Node JS setup

To verify the installation, run “***npm -v***” in the terminal.

Some common options in the **Node CLI** are :-

- **-v** : gets the version of Node JS
- **-e** : evaluates the argument string
- **-c** : check the syntax of a file

```
PS C:\Users\Mojo_Ji> node -v
v16.16.0
PS C:\Users\Mojo_Ji> node -e 'console.log(3 + 2)'
5
```

NPM

NPM stands for Node Package Manager. It is a tool that aids you in installing and managing the Node JS libraries into your project.

NPM offers both **local** and **global** installation of the libraries.

By default, libraries are installed locally. For global installation use,

npm install -g *package-name*

NPM

To setup a node project, run “npm init -y” in the terminal.

This creates a ***package.json*** file into the project folder, which keeps track of information about the project.

```
PS C:\Users\Mojo_Ji\Desktop\demo_project> npm -v
8.11.0
PS C:\Users\Mojo_Ji\Desktop\demo_project> npm init --y
Wrote to C:\Users\Mojo_Ji\Desktop\demo_project\package.json:

{
  "name": "demo_project",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}


PS C:\Users\Mojo_Ji\Desktop\demo_project>
```

Installation of dependencies

The install of a dependency into the project results in creation of a ***node_modules*** folder and ***package-lock.json*** file. The *package.json* file automatically gets updated.

```
up to date, audited 58 packages in 772ms

7 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\Mojo_Ji\Desktop\demo_project> npm install express
[                    ] / idealTree: timing idealTree Completed in 157ms
```

```
12    "dependencies": {
13      "express": "^4.18.1"
14    },
15    "devDependencies": {
16      "morgan": "^1.10.0"
17    }
```

Some commands commonly used to install dependencies :-

- **npm install *package-name*** : installs a package into your project.
- **npm uninstall *package-name*** : removes a package from your project.
- **npm install -D *package-name*** : installs package as a *dev-dependency*.

Sample package.json file

package.json × index.js config.js

package.json > {} scripts > config

```
1  {
2    "name": "demo_project",
3    "version": "1.0.0",
4    "description": "this project serves as a demo for learning
     purpose in the class",
5    "main": "index.js",
       Debug
6    "scripts": {
7      "prod": "node index.js",
8      "dev": "nodemon index.js",
9      "config": "node config.js"
10   },
11   "keywords": [],
12   "author": "",
13   "license": "ISC",
14   "dependencies": {
15     "express": "^4.18.1"
16   },
17   "devDependencies": {
18     "morgan": "^1.10.0",
19     "nodemon": "^2.0.19"
20   }
21 }
22
```

EXPLORER

DEMO_PROJECT

- node_modules
- config.js
- index.js
- package-lock.json
- package.json

OUTLINE

TIMELINE

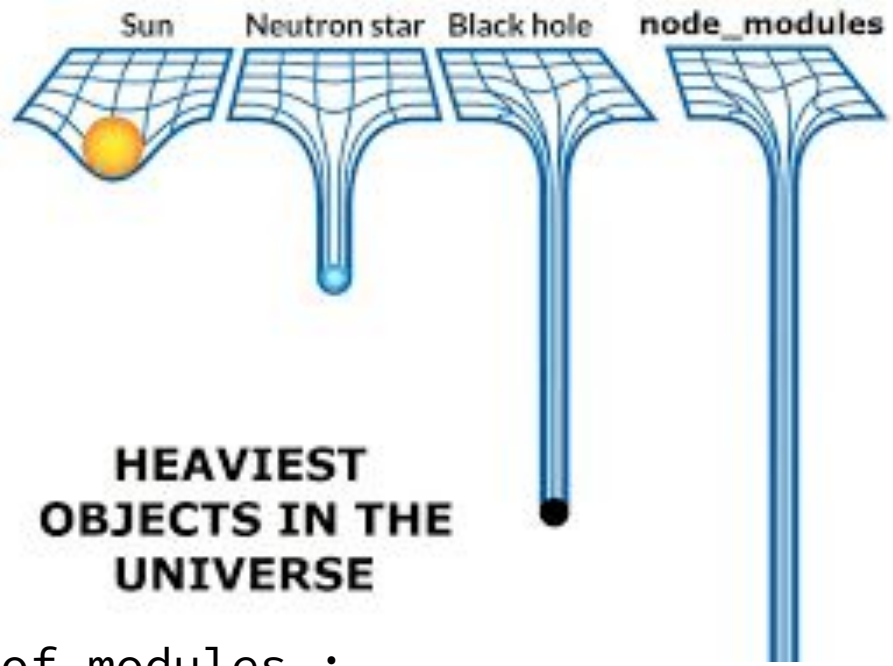
NPM SCRIPTS

Node JS Global Object

Node.js global objects are universal in nature and available in all modules.
Some commonly used are as follows :

- **__filename** represents the filename of currently executing script.
- **__dirname** represents the name of directory of currently executing script.
- **setTimeout(*callback, timeout*)** is used to run *callback* after at least *timeout* milliseconds.
- **setInterval(*callback, interval*)** is used to run *callback* repeatedly after at least *interval* milliseconds.
- **Console** is used to print information on stdout and stderr.

Module in node



Module is a simple functionality organized in multiple JS files which can be reused.

Node JS includes following types of modules :

- **Core Modules** include the main functionalities of Node JS. Some frequently used are :- http, url, path, fs etc.

Module in node

- **Local Modules** are modules created locally by the developer. These are custom objects which can be distributed in separate files and folders.

log_module.js U ×

modules > log_module.js > <unknown>

```javascript
var log = {
    info: function (info) {
        console.log('Info: ' + info);
    },
    warning:function (warning) {
        console.log('Warning: ' + warning);
    },
    error:function (error) {
        console.log('Error: ' + error);
    }
};

module.exports = log
```

detail_module.js U ×

modules > detail_module.js > <unknown> > getName

```javascript
function getName() {
    return "Mojo";
};

function getLocation() {
    return "Prayagraj";
};

const dateOfBirth = "26.03.2001";

module.exports = { getName, getLocation, dateOfBirth };
```

Module in node

index.js M ×

index.js > ...

You, 8 seconds ago | 1 author (You)

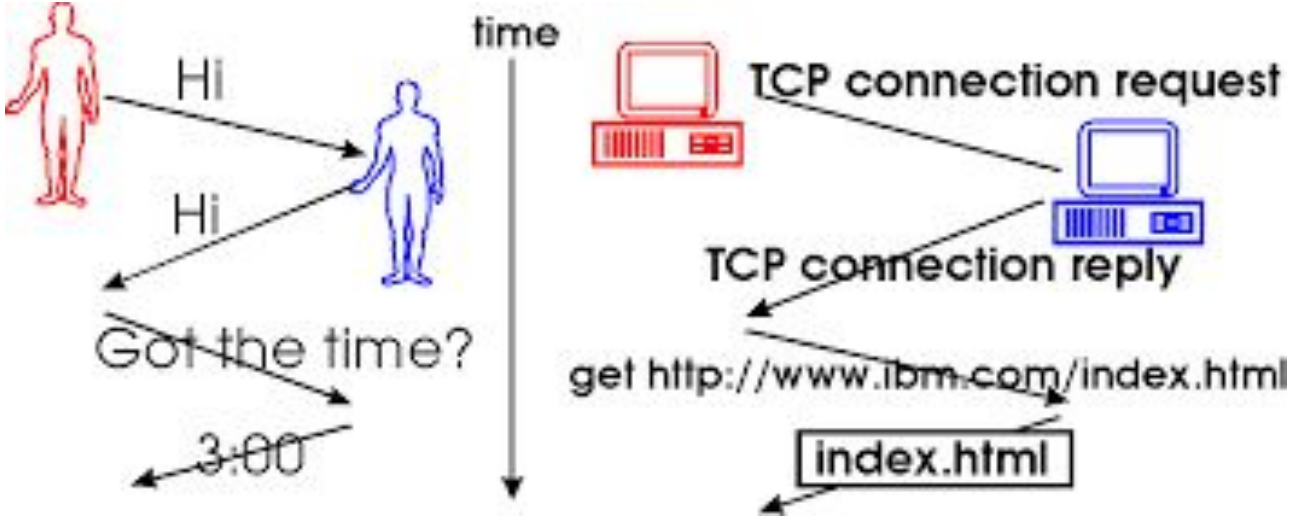
```js
var log = require("./modules/log_module");
var { getName, getLocation, dateOfBirth } = require("./modules/detail_module");

log.info(getName());
log.info(getLocation());
log.info(dateOfBirth);
log.error("user not found!!");
```

```
Mojo_Ji@mojo-windows MINGW64 ~/Desktop/demo_project (master)
$ npm run prod

> demo_project@1.0.0 prod
> node index.js

Info: Mojo
Info: Prayagraj
Info: 26.03.2001
Error: user not found!!
```

Network Fundamentals

The **Internet** is a vast network that connects computers all over the world.

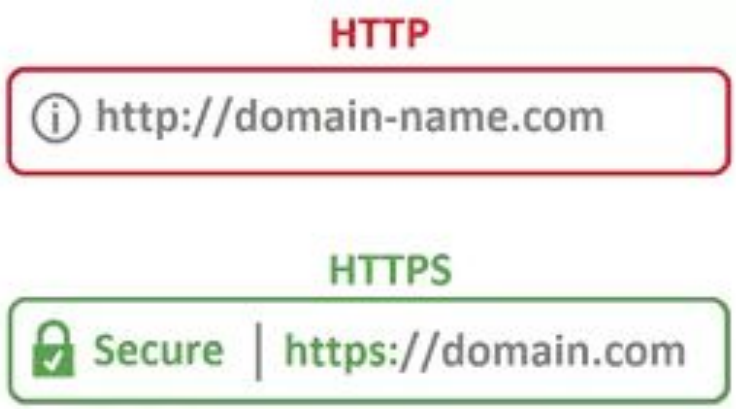
A **Protocol** is a system of rules that allows two or more computers to communicate in the Internet.



Network Fundamentals

HTTP or HyperText Transfer Protocol is a protocol for transmitting hypermedia documents such as HTML, audio, video etc. through web to the browser.

HTTPS is a secure extension of HTTP protocol.



HTTP Request/Response

The node server catches the requests and converts them into JS object.

Headers are fields which provide information about the request and the client.

```
POST / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macintosh;… )… Firefox/51.0
Accept: text/html,application/xhtml+xml,…,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=-12656974
Content-Length: 345

-12656974
(more data)
```

Request headers

General headers

Representation headers

HTTP Request/Response

Methods indicate the desired action to be performed for a given resource. Example : **GET**, **POST** etc.

Response Code indicate whether a specific request has been successfully completed.

Request/Response Body contains the information to be exchanged between the server and the client.

HTTP STATUS CODES

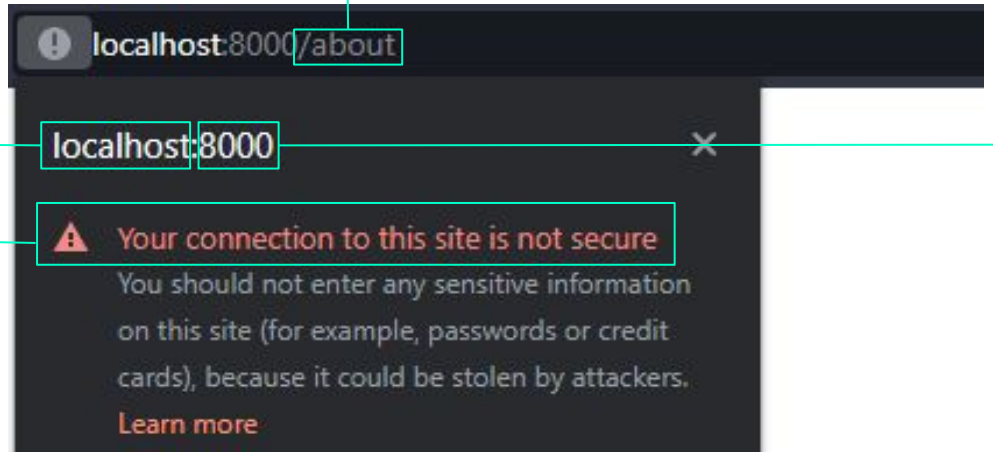
2xx Success	
200	Success / OK
3xx Redirection	
301	Permanent Redirect
302	Temporary Redirect
304	Not Modified
4xx Client Error	
401	Unauthorized Error
403	Forbidden
404	Not Found
405	Method Not Allowed
5xx Server Error	
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout

URLs

URL stands for Uniform Resource Locator.

Path indicates the location of the resource or information at the server that client wants to access.

Hostname of the HTTP server



Port Number mapped with server process in the host computer.

Indicates HTTP (Not Secure) Request

Static web pages

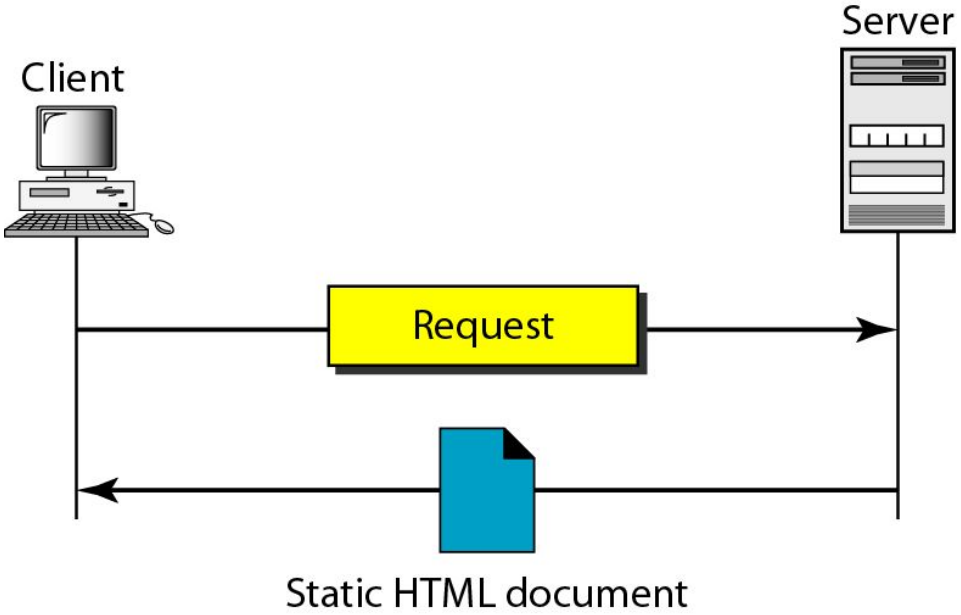
The contents of web page remains same for every client.
Suitable for never or rarely updated content.

Advantages :

- Quick and Cheap to develop.
- Cheap to host.
- Loads very quick.

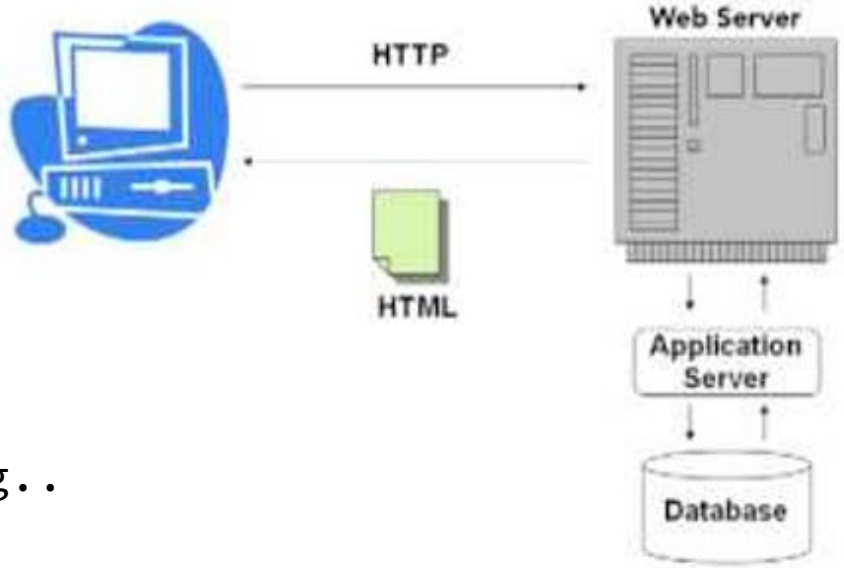
Disadvantages :

- Requires a web developer to modify.



Dynamic web pages

The contents are customized according the requesting user.
The server creates a fresh **document** for each incoming request.



Advantages :

- Easy to restructure and manage.
- Use of database as a store.

Disadvantages :

- Take longer to load.
- More work and cost in designing..

Active web pages

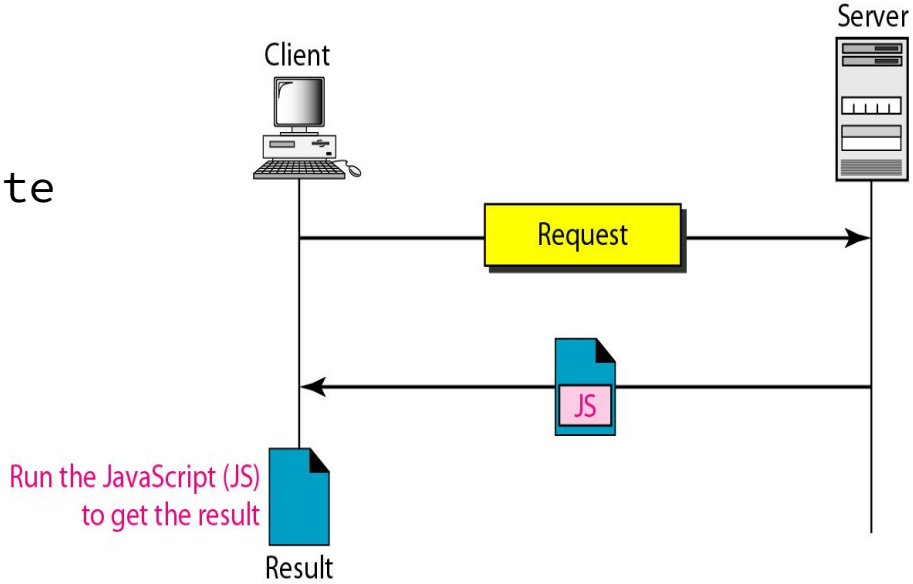
The computer program (JS or applet) is sent to the client browser to run locally, which interacts and changes display continuously.

Advantages :

- Ability to access sources of information directly and update web page continuously.

Disadvantages :

- Program should be platform independent.
- Potential risk.



Creating An HTTP Server

index.js M ×

index.js > server > http.createServer() callback

You, now | 1 author (You)

```
1  var http = require("http");
2  var assets_module = require("./modules/assets_module");
3  var log_module = require("./modules/log_module")
4
5  const PORT = 8000;
6
7  const server = http.createServer(function(req, res) {
8      let { httpHeaders, httpUrl, httpMethod } = req.url;
9      console.log(httpHeaders);        You, now • Uncommitted changes
10
11     switch(httpUrl) {
12         case "/":
13             render(res, "index.html");
14             break;
15         case "/about":
16             render(res, "about.html");
17             break;
18         case "/services":
19             render(res, "index.html");
20             break;
21         default:
22             render(res, "not_found.html");
23     }
24
25     log_module.info(`${httpMethod} ${httpUrl} | STATUS : ${res.statusCode}`);
26 });
```

When an HTTP request hits the server, node calls this request handler function with the **req:REQUEST** incoming object.

Switch case is used to separate the server response in accordance with the ***url*** and ***method*** in the HTTP request. Also called **routing.**

Continued...

index.js M ×

index.js > server > http.createServer() callback

```
28  function render(res, htmlFile) {
29      assets_module.getFileContent(`./${htmlFile}`, (err, content) => {
30          if(err) {
31              log_module.error(err);
32              res.statusCode = 500;
33              res.setHeader('Content-Type', 'text/html');
34              res.end('Some Internal Error Occured!');
35          }
36          else {
37              res.statusCode = 200;
38              res.setHeader('Content-Type', 'text/html');
39              res.end(content);
40          }
41      })
42  }
43
44  server.listen(PORT, () => {
45      log_module.info(`server listening at port ${PORT}`);
46  });
```

Customizing the **response** by setting the status code, headers and body of the response.

Starting the infinite loop which is constantly listening to any incoming connections

Continued...

Core Modules

A string which indicates the type of file sent with the response body

assets_module.js M

modules > assets_module.js > assets_module > getExtName

You, 9 hours ago | 1 author (You)

```javascript
var path = require("path");
var fs = require("fs");

const mimeTypes = {
    '.html': 'text/html',
    '.js': 'text/javascript',
    '.css': 'text/css',
    '.json': 'application/json',
    '.png': 'image/png',
    '.jpg': 'image/jpg',
};

var assets_module = {
    getExtName : function(filePath) {        You, 9 hours ago • http server and routing
        return String(path.extname(filePath)).toLowerCase();
    },
    getMimeType : function(fileExt) {
        return mimeTypes[fileExt] ==/== undefined ? mimeTypes[fileExt] : 'application/octet-stream'
    },
    getFileContent : function(filePath, callback) {
        let fileLocation = path.join(__dirname, "../assets", filePath);

        fs.readFile(fileLocation, (err, content) => {
            if(err) console.log(err);
            callback(err, content);
        })
    }
};

module.exports = assets_module;
```

Reads the contents of the ***filePath*** and initiates ***callback*** which sends ***content*** as the body of the response object.

THANK YOU!!