

NODE.JS CC NOTES

Path Module

```
JS logger.js    JS path_demo.js X    JS os_demo.js    JS index.js
reference > JS path_demo.js > ...
1  // working with path module
2
3  const path = require('path');
4
5  // Base file name
6  console.log(path.basename(__filename));
7
8
9  // Directory name
10 console.log(path.dirname(__filename)); // or we can use __dirname instead of using path
11
12
13 // File Extension
14 console.log(path.extname(__filename));
15
16
17 // Create path object
18 console.log(path.parse(__filename));
19
20 console.log(path.parse(__filename).base);
21
22
23 // Concatenate path
24 // ../ test/ hello.html
25 console.log(path.join(__dirname, 'test', 'hello.html'));
```

```
(base) anuragjanghala@ajpc:~/Desktop/nodejs_CC$ cd reference
(base) anuragjanghala@ajpc:~/Desktop/nodejs_CC/reference$ node path_demo
path_demo.js
/home/anuragjanghala/Desktop/nodejs_CC/reference
.js
{
  root: '/',
  dir: '/home/anuragjanghala/Desktop/nodejs_CC/reference',
  base: 'path_demo.js',
  ext: '.js',
  name: 'path_demo'
}
path_demo.js
/home/anuragjanghala/Desktop/nodejs_CC/reference/test/hello.html
(base) anuragjanghala@ajpc:~/Desktop/nodejs_CC/reference$
```

OS MODULE

```
JS logger.js      JS os_demo.js ×      JS url_demo.js
reference > JS os_demo.js > ...
 1  const os = require('os');
 2
 3  // platform
 4  console.log(os.platform());
 5
 6
 7  // CPU Arch
 8  console.log(os.arch())
 9
10
11  // CPU core info
12  console.log(os.cpus());
13
14
15  // Free memory
16  console.log(os.freemem());
17
18
19  // total memory
20  console.log(os.totalmem());
21
22
23  // Home dir
24  console.log(os.homedir());
25
26
27  // Uptime
28  console.log(os.uptime());
```

```
(base) anuragjanghala@ajpc:~/Desktop/nodejs_CC/reference$ node os_demo
linux
x64
[
  {
    model: 'Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz',
    speed: 3984,
    times: { user: 204810, nice: 2680, sys: 82750, idle: 5021030, irq: 0 }
  },
  {
    model: 'Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz',
    speed: 3980,
    times: { user: 217850, nice: 4730, sys: 81800, idle: 4951540, irq: 0 }
  },
  {
    model: 'Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz',
    speed: 3972,
    times: { user: 219940, nice: 2300, sys: 80610, idle: 4986700, irq: 0 }
  },
  {
    model: 'Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz',
    speed: 3966,
    times: { user: 195560, nice: 950, sys: 89970, idle: 4994820, irq: 0 }
  },
  {
    model: 'Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz',
    speed: 3985,
    times: { user: 210450, nice: 1740, sys: 81800, idle: 5009580, irq: 0 }
  },
  {
    model: 'Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz',
    speed: 3997,
    times: { user: 206800, nice: 3330, sys: 83430, idle: 5014340, irq: 0 }
  },
  {
    model: 'Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz',
    speed: 3997,
    times: { user: 210610, nice: 2780, sys: 90490, idle: 5001070, irq: 0 }
  },
  {
    model: 'Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz',
    speed: 3967,
    times: { user: 210520, nice: 2700, sys: 79800, idle: 5024160, irq: 0 }
  },
  {
    model: 'Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz',
    speed: 3946,
    times: { user: 210620, nice: 2660, sys: 80530, idle: 5014070, irq: 0 }
  },
  {
    model: 'Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz',
    speed: 3998,
    times: { user: 215970, nice: 1620, sys: 80220, idle: 5004810, irq: 0 }
  },
  {
    model: 'Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz',
    speed: 3993,
    times: { user: 211756, nice: 1616, sys: 81852, idle: 501756, irq: 0 }
  }
]
1317564416
8185262080
/home/anuragjanghala
5367
```

FS Module

```
JS logger.js    JS fs_demo.js X    JS os_demo.js    JS index.js
reference > JS fs_demo.js > ...
1  const fs = require('fs');
2  const path = require('path');
3
4  // Create folder
5  fs.mkdir(path.join(__dirname, '/test'), {}, err => {
6    if(err) throw err;
7    console.log('Folder created...');
8  });
9
10 // Create and write to file
11 fs.writeFile(path.join(__dirname, '/test', 'hello.txt'), 'Hello World!', err => {
12   if(err) throw err;
13   console.log('Folder written to...');
14
15   fs.appendFile(path.join(__dirname, '/test', 'hello.txt'), 'I love node.js', err => {
16     if(err) throw err;
17     console.log('Folder written to...');
18   });
19 });
20
21
22 // this will overwrite
23 fs.writeFile(path.join(__dirname, '/test', 'hello.txt'), 'I love node.js', err => {
24   if(err) throw err;
25   console.log('Folder written to...');
26 });
27
28
29 // Read File
30 fs.readFile(path.join(__dirname, '/test', 'hello.txt'), 'utf-8', (err,data) => {
31   if(err) throw err;
32   console.log(data);
33 });
34
35 // Rename File
36 fs.rename(
37   path.join(__dirname, '/test', 'hello.txt'),
38   path.join(__dirname, '/test', 'helloworld.txt'),
39   err => {
40     if(err) throw err;
41     console.log('File renamed...');
42   }
43 );
```

Event Emitter using log

-> uuid was used to get the id of the message

```
JS logger.js x JS index.js JS fs_demo.js JS os_demo.js
JS logger.js > <unknown>
1  const EventEmitter = require('events');
2  const uuid = require('uuid');
3
4  class Logger extends EventEmitter{
5      log(msg){
6          // Call event
7          this.emit('message', {id : uuid.v4(), msg});
8      }
9  }
10
11 module.exports = Logger
```

```
JS logger.js JS index.js x JS fs_demo.js JS os_demo.js
JS index.js > logger.on('message') callback
1  const Logger = require('./logger');
2
3  const logger = new Logger();
4
5  logger.on('message', (data) => console.log('called listener', data));
6
7  logger.log('Hello world!');
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 3: bash
(base) anuragjanghala@ajpc:~/Desktop/nodejs_CC$ cd reference
(base) anuragjanghala@ajpc:~/Desktop/nodejs_CC/reference$ cd ..
(base) anuragjanghala@ajpc:~/Desktop/nodejs_CC$ node index
called listener { id: 'de4293e9-0533-413e-9364-0fc4dead2e21', msg: 'Hello world!' }
(base) anuragjanghala@ajpc:~/Desktop/nodejs_CC$
```

Classes using export modules:

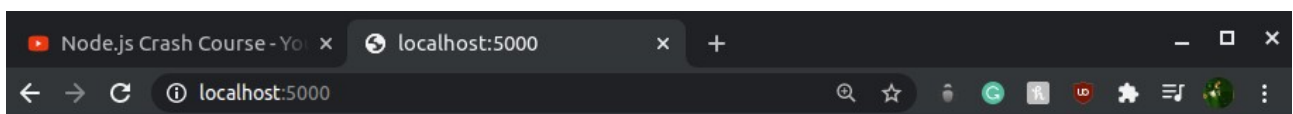
```
JS person.js X
JS person.js > Person > greeting
1  class Person {
2      constructor(name, age){
3          this.name = name;
4          this.age = age;
5      }
6
7      greeting(){
8          console.log("My name is " +this.name+ " and I am "+this.age);
9      }
10 }
11
12
13
14 module.exports = Person;
```

HTTP Module Server

- Simplest web server -> hello world

```
JS http_demo.js x
reference > JS http_demo.js > ...
1  const http = require('http');
2
3  // Create server object
4  http.createServer((req, res) => { //request and response
5      // Write a response
6      res.write('Hello WOrld!');
7      res.end();
8  }).listen(5000, () => console.log('Server running...'));
```

```
(base) anuragjanghala@ajpc:~/Desktop/nodejs_CC/reference$ node http_demo
Server running...
^C
(base) anuragjanghala@ajpc:~/Desktop/nodejs_CC/reference$ cd ..
```



Hello World!

Nodemon :

is used to watch over the running server so that we need not to update server on our own.

```
JS index.js    {} package.json X
{} package.json > ...
1  {}
2  "name": "nodejs_cc",
3  "version": "1.0.0",
4  "description": "Node crash course",
5  "main": "index.js",
6  > Debug
7  "scripts": {
8    "start": "node index",
9    "dev": "nodemon index"
10 },
11 "author": "Anurag Janghala",
12 "license": "ISC",
13 "dependencies": {
14   "uuid": "^8.3.1"
15 },
16 "devDependencies": {
17   "nodemon": "^2.0.6"
18 }
19 }
```

```
JS index.js X    {} package.json
JS index.js > [?] server > [?] http.createServer() callback
1  const http = require('http');
2  const path = require('path');
3  const fs = require('fs');
4
5
6  const server = http.createServer((req,res) => {
7    if (req.url === '/') {
8      res.end('<h1>HomeLander</h1>');
9    }
10  });
11
12  const PORT = process.env.PORT || 5001;
13
14  server.listen(PORT, () => console.log(`Server running on port `+ PORT));
```



```
(base) anuragjanghala@ajpc:~/Desktop/nodejs_CC$ npm run dev
> nodejs_cc@1.0.0 dev /home/anuragjanghala/Desktop/nodejs_CC
> nodemon index

[nodemon] 2.0.6
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index index.js`
Server running on port 5001
[nodemon] restarting due to changes...
[nodemon] starting `node index index.js`
Server running on port 5001
█
```

Full Server code using nodeJS and Some html, css files used to deploy on heroku using git.

```
JS index.js x # styles.css <> about.html <> index.html <> 404.html
JS index.js > [?] server > http.createServer() callback
1  const http = require('http');
2  const path = require('path');
3  const fs = require('fs');
4
5
6  const server = http.createServer((req, res) => {
7
8      // // for calling the normal webpages
9      // if (req.url === '/') {
10         // fs.readFile(
11             //     path.join(__dirname, 'public', 'index.html'),
12             //     (err, content) => {
13                 // if (err) throw err;
14                 // res.writeHead(200, {'Content-Type': 'text/html'});
15                 // res.end(content);
16             // }
17         // );
18     // }
19     // if (req.url === '/about') {
20         // fs.readFile(
21             //     path.join(__dirname, 'public', 'about.html'),
22             //     (err, content) => {
23                 // if (err) throw err;
24                 // res.writeHead(200, {'Content-Type': 'text/html'});
25                 // res.end(content);
26             // }
27         // );
28     // }
```

```
// // for calling the apis
// if (req.url === '/api/users') {
//     const users = [
//         { name: 'Bob Smith', age: 40 },
//         { name: 'John Doe', age: 30 }
//     ];
//     res.writeHead(200, {'Content-Type': 'application/json'});
//     res.end(JSON.stringify(users));
// }

let filePath = path.join(
    __dirname,
    'public',
    req.url === '/' ? 'index.html' : req.url
);

// extension of the file
let extname = path.extname(filePath);

// Initial content type
let contentType = 'text/html';
```

```

//check extension and set content type
switch(extname){
  case '.js':
    contentType = 'text/javascript';
    break;
  case '.css':
    contentType = 'text/css';
    break;
  case '.json':
    contentType = 'application/json';
    break;
  case '.png':
    contentType = 'image/png';
    break;
  case '.jpg':
    contentType = 'image/jpg';
    break;
}

```

```

// Read file
fs.readFile(filePath,
  (err, content) => {
    if(err){
      if(err.code == 'ENOENT'){
        // Page not found
        fs.readFile(path.join(__dirname, 'public', '404.html'), (err, content) =>{
          if(err) throw err;
          res.writeHead(200, {'Content-Type': 'text/html'});
          res.end(content, 'utf-8');
        })
      }else {
        // Some server error
        res.writeHead(500);
        res.end(`Server error: `+err.code);
      }
    }else {
      // Success
      res.writeHead(200, {'Content-Type': contentType});
      res.end(content, 'utf-8');
    }
  }
);

const PORT = process.env.PORT || 5001;

server.listen(PORT, () => console.log(`Server running on port `+ PORT));

```

Html files:

```
public > <> index.html > html > head > link
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>Homepage</title>
7      <link rel="stylesheet" href="./css/styles.css" />
8    </head>
9    <body>
10     <h1>Welcome to the Homepage!</h1>
11   </body>
12 </html>
13
```

```
JS index.js # styles.css <> about.html x <> index.html <> 404.html
public > <> about.html > html > head > link
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>About</title>
7      <link rel="stylesheet" href="./css/styles.css" />
8    </head>
9    <body>
10     <h1>About</h1>
11   </body>
12 </html>
13
```

```
JS index.js # styles.css <> about.html <> index.html <> 404.html x
public > <> 404.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>error 404</title>
7      <link rel="stylesheet" href="./css/styles.css" />
8    </head>
9    <body>
10     <h1>404 Error</h1>
11   </body>
12 </html>
13
```

CSS file:

```
JS index.js # styles.css X <> abo
public > css > # styles.css > body
1 body {
2     background: #333;
3     color: #fff;
4 }
```