Unit – 3 Node JS

Node JS File System

- The Node.js file system module allows you to work with the file system on your server.
- We use this module for file various operations like creating, reading, deleting, updating file etc.,
- Node.js gives the functionality of file I/O by providing wrappers functions around the standard POSIX functions.
- All file system related function can be synchronous and asynchronous depending upon user requirements.
- There are multiple ways to work with file.

- **What is Synchronous approach?**
- In **synchronous** approach, suppose you call FunctionA() and then FunctionB(), in this case first FunctionA() will Complete it’e execution then only FunctionB() will execute.
- **What is asynchronous approach?**
- In **asynchronous** approach, suppose you call FunctionA() and then FunctionB(), in this case FunctionA() will start first but FunctionB() will also start.
- FunctionB() will not wait for FunctionA() to complete it’s execution.

Common File operations are

- Read Files
- Write Files
- Append Files
- Open Files
- Close Files
- Rename Files
- Delete Files

How to use file module

- To use file module we need to include the File System module useing the require() method:
- var fs = require('fs');

How to read data from file?

- First create file in project folder. For example myfile.html and write some content in it.
- The fs.readFile() method is used to read files on your computer in this case myfile.html. It works **asynchronously.**
- This method has 2 argument. 1st argument is filename to be read and 2nd argument is anonymous function that will run once when content is fetched from file.
- This function also has 2 argument, 1st argument is error and 2nd argument is data read from file.
- Let us see an example.

Example of reading data from file asynchronously?

```
var http = require('http');
var fs = require('fs');
var server = http.createServer(function (request, response) {
  // Asynchronous read
  //'myfile.html' must exist in same directory where this file is
  fs.readFile('myfile.html', function(error,FileContent)
  {
    response.writeHead(200, {'Content-Type': 'text/html'});
    response.write(FileContent);
    return response.end();
  });
});
server.listen(5000)
```

How to load html file dynamically

how to load html file dynamically

```
var http = require('http');
var url = require('url');
var fs = require('fs');
// http://127.0.0.1:5000/mango.html
// http://127.0.0.1:5000/apple.html
http.createServer(function (request,response) {
  var query = url.parse(request.url, true);
  var filename = "." + query.pathname;
  console.log(filename);
  fs.readFile(filename, function(error, data) {
    if (error)
    {
      response.writeHead(404, {'Content-Type': 'text/html'});
      return response.end("404 Not Found");
    }
    response.writeHead(200, {'Content-Type': 'text/html'});
    response.write(data);
    return response.end();
  });
}).listen(5000);
```

Example of reading data from file synchronously?

```
var http = require('http');
var fs = require('fs');
var server = http.createServer(function (request, response) {
  // synchronous read

  //'myfile.html' must exist in same directory where this file is
  response.writeHead(200, {'Content-Type': 'text/html'});
  var FileContent = fs.readFileSync('myfile.html');
  response.write(FileContent);

  return response.end();
});
server.listen(5000);
```

How to write data into a file?

- To write data into file **asynchronously** use fs.writeFile() method.
- If file already exists then it **overwrites** the existing content otherwise it creates a **new** file and writes data into it.
- It has 4 arguments.
  - 1st argument is fillname.
  - 2nd argument is data to be written into file.
  - 3rd argument is optional, it include encoding, mode and flag.
  - 4th argument is callback function which will execute automatically after content is written into file.

Example of writing data into file

Writing data into file

```
var fs = require('fs');
var FileContent = "I like banana. it is both healthy and testy"
fs.writeFile('banana.txt',FileContent, function (error) {
    if (error)
        console.log(error);
    else
        console.log('Content is Written into file successfully');
});
```

How to append (add new data) into existing file?

- To appned data into file asynchronously use fs.appendFile() method.
- If file already exists then it **add new content in** the existing content otherwise it creates a **new** file and writes data into it.
- It has 4 arguments.
  - 1st argument is fillname.
  - 2nd argument is data to be written into file.
  - 3rd argument is optional, it include encoding, mode and flag.
  - 4th argument is callback function which will execute automatically after content is written into file.

Example of appending data into file

append data into file

```
var fs = require('fs');
var FileContent = "\nBanana has yellow color. and it is usually of 6 to 8 inch long."
fs.appendFile('banana.txt',FileContent, function (error) {
    if (error)
        console.log(error);
    else
        console.log('Content is added into file successfully');
});
```

How to write data into a file synchronously ?

- To write data into file synchronously use fs. WriteFileSync() method.
- If file already exists then it **overwrite** existing content otherwise it creates a **new** file and writes data into it.
- It has 4 arguments.
  - 1st argument is fillname.
  - 2nd argument is data to be written into file.
  - 3rd argument is optional, it include encoding, mode and flag.

Example of writing data into file asynchronously

append data into file asynchronosly

```
var fs = require('fs');
var FileContent = "apple banana mango pinapple orange\n";
fs.appendFileSync('fruits.txt', FileContent, 'utf8')
console.log('file create/updated successfully');
```

How to delete existing file?

- To delete file we use fs.unlink() method.
- Syntax
- fs.unlink(path, callback)
  1. 1st argument in this function is path and file name
  2. 2nd argument is callback function that will execute after file gets deleted.

Example of how to delete file?

how to delete file

```
var fs = require('fs');
fs.unlink('fruits.txt', function (error) {
    if (error)
        console.log('file could not be deleted.')
    else
        console.log('file deleted sucessfully');
});
```

How to rename file?

- To rename a file with the File System module, use the fs.rename() method.
- The fs.rename() method has 3 arguments.
  1. 1st argument is current file name
  2. 2nd argument is new file name
  3. 3rd argument is callback function that will execute after file is renamed.

Example of how to rename file

how to rename file

```
var fs = require('fs');
fs.rename('banana.txt', 'pinaaple.txt', function (err) {
  if (err)
    console.log('File cound not be renamed!');
  else
    console.log('File Renamed sucessfully');
});
```