Node.js (server-side JavaScript)

Node.js NPM

- NPM is a package manager for Node.js packages, or modules if you like.
- NPM comes bundled with Node.js installables after v0.6.3 version. To verify the same, open console and type the following command and see the result −
- $ npm –version

- **Installing Modules(Package) using NPM –**

- There is a simple syntax to install any Node.js module −
- $ npm install <Module Name>
- For Example - I want to download a package called "uppercase":
- C:\Users\Your Name>npm install uppercase

- Now you have downloaded and installed your first package!
- NPM creates a folder named "node_modules", where the package will be placed. All packages you install in the future will be placed in this folder.
- **Using a Package -**
- Once the package is installed, it is ready to use.
- Include the "upper-case" package the same way you include any other module:
- var uc = require('uppercase');
- Create a Node.js file that will convert the output "Hello World!" into upper-case letters:

- Example

```
var http = require('http');
var uc = require('uppercase');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type':
    'text/html'});
  res.write(uc("Hello World!"));
  res.end();
}).listen(8080);
```

- **Uninstalling a Module**
- Use the following command to uninstall a Node.js module.
- $ npm uninstall upper-case
- Once NPM uninstalls the package, you can verify it by looking at the content of /node_modules/ directory or type the following command –
- $ npm ls

- **Updating a Module -**
- Update package.json and change the version of the dependency to be updated and run the following command.
- $ npm update upper-case

Node.js File System Module

- **Node.js as a File Server**
- The Node.js file system module allows you to work with the file system on your computer.
- To include the File System module, use the require() method:
- var fs = require('fs');
- **Common use for the File System module:**
- Read files
- Create files
- Update files
- Delete files
- Rename files

- **Read Files**
- The fs.readFile() method is used to read files on your computer.
- Assume we have the following HTML file (located in the same folder as Node.js):-
- **demofile1.html -**

```
<html>
<body>
<h1>My Header</h1>
<p>My paragraph.</p>
</body>
</html>
```

- Create a Node.js file that reads the HTML file, and return the content:
- Example

```
var http = require('http');
var fs = require('fs');
http.createServer(function (req, res) {
  fs.readFile(‘demofile1.html', function(err, data) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(data);
    res.end();
  });
}).listen(8080);
```

- Save the code above in a file called "demo_readfile.js", and initiate the file:
- Initiate demo_readfile.js:
- C:\Users\Your Name>node demo_readfile.js
- If you have followed the same steps on your computer, you will see the same result as the example: http://localhost:8080

- **Create Files**
- The File System module has methods for creating new files:
- **fs.appendFile()**
- **fs.open()**
- **fs.writeFile()**

1- The fs.appendFile() method appends specified content to a file. If the file does not exist, the file will be created:

- Example
- Create a new file using the appendFile() method:

```
var fs = require('fs');

fs.appendFile('mynewfile1.txt', 'Hello content!', function
    (err) {
  if (err) throw err;
  console.log('Saved!');
});
```

2- The fs.open() method takes a "flag" as the second argument, if the flag is "w" for "writing", the specified file is opened for writing. If the file does not exist, an empty file is created:

- Example
- Create a new, empty file using the open() method:

```
var fs = require('fs');

fs.open('mynewfile2.txt', 'w', function (err, file) {
  if (err) throw err;
  console.log('Saved!');
});
```

3- The fs.writeFile() method replaces the specified file and content if it exists. If the file does not exist, a new file, containing the specified content, will be created:

- Example
- Create a new file using the writeFile() method:

```
var fs = require('fs');

fs.writeFile('mynewfile3.txt', 'Hello content!',
    function (err) {
  if (err) throw err;
  console.log('Saved!');
});
```

- **Delete Files**
- To delete a file with the File System module, use the fs.unlink() method.
- The fs.unlink() method deletes the specified file:
- Example
- Delete "mynewfile2.txt":

```
var fs = require('fs');

fs.unlink('mynewfile2.txt', function (err) {
  if (err) throw err;
  console.log('File deleted!');
});
```

- **Rename Files**
- To rename a file with the File System module, use the fs.rename() method.
- The fs.rename() method renames the specified file:
- Example
- Rename "mynewfile1.txt" to "myrenamedfile.txt":
- var fs = require('fs');

  ```
  fs.rename('mynewfile1.txt', 'myrenamedfile.txt', function (err) {
    if (err) throw err;
    console.log('File Renamed!');
  });
  ```