REST & File System module

Objective

- Have a basic understanding of REST architecture.
- Know how to use File System module to manipulate with files.

Agenda

- RESTful API
- RESTful Web Service
- File System module

RESTful API

- REST stands for REpresentational State Transfer. REST is web standards based architecture and uses HTTP Protocol. It revolves around resource where every component is a resource and a resource is accessed by a common interface using HTTP standard methods. REST was first introduced by Roy Fielding in 2000

REST

- REST uses various representation to represent a resource like text, JSON, XML but *JSON* is the most popular one

HTTP methods

Following four HTTP methods are commonly used in REST based architecture.

- **GET** - This is used to provide a read only access to a resource.
- **PUT** - This is used to create a new resource.
- **DELETE** - This is used to remove a resource.
- **POST** - This is used to update an existing resource or create a new resource.

RESTful Web Service

- A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks.

- Web services based on REST Architecture are known as RESTful web services. These web services uses HTTP methods to implement the concept of REST architecture. A RESTful web service usually defines a URI, Uniform Resource Identifier a service, which provides resource representation such as JSON and set of HTTP Methods.

Exercise (http module & Rest)

S. N.	URI	HTTP Method	POST body	Result
1	listUsers	GET	empty	Show list of all the users.
2	addUser	POST	JSON String	Add details of new user.
3	deleteUser	DELETE	JSON String	Delete an existing user.
4	:id	GET	empty	Show details of a user.

File System

- Node implements File I/O using simple wrappers around standard POSIX functions. The Node File System (fs) module can be imported using the following syntax

How to use

- var fs = require("fs")
- Every method in the fs module has synchronous as well as asynchronous forms

How to open a file

Open a File

- fs.open(path, flags[, mode], callback)
- Here is the description of the parameters used −
  - **path** − This is the string having file name including path.
  - **flags** − Flags indicate the behavior of the file to be opened. All possible values have been mentioned below.
  - **mode** − It sets the file mode (permission and sticky bits), but only if the file was created. It defaults to 0666, readable and writeable.
  - **callback** − This is the callback function which gets two arguments (err, fd).

How to open a file

Flag	Description
r	Open file for reading. An exception occurs if the file does not exist.
r+	Open file for reading and writing. An exception occurs if the file does not exist.
rs	Open file for reading in synchronous mode.
rs+	Open file for reading and writing, asking the OS to open it synchronously. See notes for 'rs' about using this with caution.
w	Open file for writing. The file is created (if it does not exist) or truncated (if it exists).
wx	Like 'w' but fails if the path exists.
w+	Open file for reading and writing. The file is created (if it does not exist) or truncated (if it exists).
wx+	Like 'w+' but fails if path exists.
a	Open file for appending. The file is created if it does not exist.
ax	Like 'a' but fails if the path exists.
a+	Open file for reading and appending. The file is created if it does not exist.
ax+	Like 'a+' but fails if the the path exists.

How to write a file

Writing a File

- fs.writeFile(path, data[, options], callback)
  - **path** − This is the string having the file name including path.
  - **data** − This is the String or Buffer to be written into the file.
  - **options** − The third parameter is an object which will hold {encoding, mode, flag}. By default. encoding is utf8, mode is octal value 0666. and flag is 'w'
  - **callback** − This is the callback function which gets a single parameter err that returns an error in case of any writing error.

How to read a file

Reading a File

- fs.read(fd, buffer, offset, length, position, callback)
- Here is the description of the parameters used −
  - **fd** − This is the file descriptor returned by fs.open().
  - **buffer** − This is the buffer that the data will be written to.
  - **offset** − This is the offset in the buffer to start writing at.
  - **length** − This is an integer specifying the number of bytes to read.
  - **position** − This is an integer specifying where to begin reading from in the file. If position is null, data will be read from the current file position.
  - **callback** − This is the callback function which gets the three arguments, (err, bytesRead, buffer).

How to delete a file

Delete a File

- fs.unlink(path, callback)
- Here is the description of the parameters used –
  - **path** – This is the file name including path.
  - **callback** – This is the callback function No arguments other than a possible exception are given to the completion callback.

Other function of fs

- Get File Information: fs.stat(path, callback)
- Closing a File: fs.close(fd, callback)
- …

Thank you