a. Details.txt , ReadFile.js and ReadFileCallback.js (Example of blocking and non-blocking code. Callback example)

1. setTimeoutProg.js (simple first example showing callback method usage in nodejs)

2. Math\_module.js,ownmoduletest.js (creating own module with variables and functions and then using it in another file and accessing the variables and functions).

3. eventEmitterProg.js (eventEmitter program demo)

4. EventEmitterModule.js , EventEmitterTest.js (same as previous but divided into a separate module 'EventEmitterModule.js' and then used in the 'EventEmitterTest.js' client js file)

5. filereadprog.js , largefile.txt (trying the fs module to read the contents from a file)

6. filereadpipetostdout.js (reading the file content and pipe to stdout)

7. filestatsprog.js (getting statistics about the file like size,atime, isfile() isdirectory etc.,)

8. processTest.js (getting details about the process like env settings, pid, current working directory etc..)

9. osmoduletest.js (getting details about the currently running system by using the os module)

10. httpServerProg.js, httpmoduletest.js (http Server and http Client component. Start the server program as "nodemon httpServerProg.js" instead of "node" . This will ensure that any changes to server program is automatically incorporated instead of again starting the program. Start the server program and minimize and then start the client program)

11. socketserverprog.js , socketclientprog1.html (start the server and then run the client html in firefox. You will see the server's response in 'logging' or 'js' tab of firefox(f12) and client's response in the server's console).

--------------------------------------EXPRESS programs--------------------------------------

under 'expressdemos' folder

12. 'demo1' folder-> expressdemo1.js (simple example of express module usage where we have created two urls and their corresponding handlers. Start 'expressdemo1.js' and access the corresponding url thru browser)

(http://localhost:3000)

(http://localhost:3000/employee)

13. 'demo2' folder -> index.js,home.js,employee.js (example having multiple routes where each route is organized in a separate javascript file. Start 'index.js' and access the corresponding url thru browser)

(http://localhost:3000/ --> calls home page)

(http://localhost:3000/employee --> calls employee index function)

(http://localhost:3000/employee/contacts --> calls employee contacts function)

14. 'demo3' folder -> index.js, employee.js (example having route params which can be passed along with request URL and how a function can retrieve it and use it)

(http://localhost:3000/employee/100 --> 100 here is the route param that can be retrieved and used in function.)

15. 'demo4' folder -> index.js, employee.js (example having query params which can be passed thru the URL in the browser and how a function can retrieve it and use it)

(http://localhost:3000/employee?id=100&name=savitha --> here two query params are sent thru the browser url id and name which will be accessed by the function and used)

16. 'demo5' folder -> index.js,views folder-home.jade,sample.jade (Trying views rendered using Jade format)

17. 'demo6' folder -> index.js,views folder-home.jade,sample.jade (Trying views

rendered using Jade format but with dynamic data supplied from js and used in jade files)

18. 'demo7' folder -> index.js, views folder, public folder (Trying views rendered using Jade format, with main layout jade file (specifying fixed view content) extended by other jade files (extending main and adding its own dynamic content) and also using specific CSS/stylus files for styling)

19. 'demo8' folder -> cookieEg.js (needs the cookie-parser module.This is a program that reates a cookie when we access the the / path and would clear the cookie when we access /clear path. (To run it: node cookie.js , then access it from the browser using localhost:3000)

20. 'demo9' folder -> sessionEg.js (needs the express-session module. This is a program hat creates a session object and puts a date into it and then we are displaying the same)

21. 'demo10' folder -> errorhandler.js, views folder:404.jade,500.jade,employee.jade (demonstrates errorhandler. If we access a valid URL then it shows the correct data, if we try to access a path that is not there then we get 404 error,if we try to access a valid URL but it does not have an appropriate rendering page, it shows 505 error. If we need to customize or create appro. jade pages that should be rendered whenever 404 or 505 errors occur, then we will create 404.jade and 505.jade which are shown when appr 404 and 505 errors occur).

22. 'demo11' folder -> interceptorEg.js (demonstrates global and single interceptors. global interceptors are called for all routes whereas single interceptors are tagged to a particular path and are called only when that url path is accessed)

23. 'demo12' folder -> localEg.js,views folder:demolocals.jade (demonstrates use of locals. like declaring global constants in JS and use it in the jade file. We have created companyName and timeData in JS and we access and use it in jade file. Run the js and then access it by saying localhost:3000/demolocals)

24.