Javascript & Microservices

Technologies

* Node.js
* Express.js
* Angular Framework
* Knockout.js
* OJET
* Spring Microservices

Pre-requisites

* HTML
* CSS
* Javascript
* Java

Software requirements

* Visual Studio Code
* Node.js
* JDK & Eclipse IDE

Full Stack - It is a concept that helps developers to create Front end and Back end applications separately so that they can make Back end applications to be reusable across many front end applications

Backend can use various technologies like

1. Java
2. Javascript
3. C#
4. Python

Frontend can use various technologies like

1. Javascript - You can use javascript for web & mobile both
2. Java - If your UI is console type (ATM, Swiping machine)
3. Python - If your UI is console type

Node.js: It is a runtime environment to run the Javascript at the backend, so that you can create applications that can access backend resources like files, OS programs, Databases and etc.

Note: Earlier Javascript was used only at the front-end and browser was the runtime environment to run the Javascript, even now also Javascript is used to create front end applications, however when you want to run the javascript to access OS programs, Databases, Files you need Node.js

Many new features were introduced in Javascript which is termed as Ecma-Script latest feature or ES6 latest feature

* Introduction of new keywords like const, let, class, extends, super
* Template literals
* Arrow functions
* Rest & Spread operators
* Object Destructuring
* Promises

let & const: These are used to create block scoped variables in place of var you use these keywords, let variables can be modified, however const variables you can’t modify

var x = 35;

let x = 35; & const x = 35;

Template string literals:

These are used to concatenate string and the javascript expressions without using + operator or without breaking the strings

You need to use back tick character to create strings & expressions.

let username = “Alex”;

let gender = “Male”;

console.log(`Name = ${username}, Gender = ${gender}`);

Earlier:

console.log(‘Name = ‘+username+’, Gender = ‘+gender);

Wrong way:

console.log(‘Name = ${username}, Gender = ${gender}’);

Arrow functions:

These are alternate way of writing callback functions in javascript (anonymous functions)

Callback functions are initiated first but called later based on some actions like events, server response and etc.

abc = function() {   
  
}

eventHandler(abc); // abc is executed when certain event occurs

Callbacks are executed later once certain event occurs like when user generates some event, or when server sends response and etc.

Callbacks are written as below

function(args) {   
  
}

Some of the inbuilt functions take callbacks as arguments which are executed on some events

ex:

arr.forEach( callbackFn )

arr.map( callbackFn )

Rest & Spread operators

Rest operator helps a function to accept 0 or more arguments

function test(…x) { }

test can accept 0 or more arguments like test(20), test(20, 10), test(30, 10, 20) and so on

Note: You can maximum have only one rest parameter & it must be the last formal meter

function test(a, …b) { } // this is fine

function test(…a, b) { } // this is wrong

Spread operator helps you to distribute the values to multiple parameters of the function

function xyz(a, b, c, d) { }

let arr = [8, 7, 2, 6];

xyz(…arr); // here a = 8, b = 7, c = 2, d = 6

Create a function that accepts 0 or more arguments and prints Maximum number, Minimum number and Sum of all the numbers in an array

ex: Input arr = [2, -1, 10, 5, 8] to the function demo(arr) then it must print

Max: 10,  
Min: -1  
Sum: 24

Note: Use a single for loop only

Node.js modules

There are reusable unit in the node project which you can create and reuse at many places

There are mainly 3 types of modules

1. Core module: Inbuilt in node.js
2. Local module: These are part of the project created by developers to reuse in their project
3. Third party module: These are the modules which you can download from the internet.

package.json: It is the main configuration file in the node.js project, it keeps project metadata like commands, libraries, dependencies

npm init -fy is used to create package.json file

By default node.js uses old version of javascript to import or export modules, hence we need to use “type”:”module” to use the new syntax of import or export

Old syntax

//export  
module.exports.demo = function() {  
}

//import

let demo = require(“./file.js”).demo;

demo();

New syntax

//export

export function demo() {  
}

//import

import {demo} from ‘./file.js’;

demo();

Core Modules

1. os
2. fs
3. http
4. url

os module: This helps you to find the os related details like platform, architecture and so on

There are many functions in os module to figure out the details of the OS like:-

1. arch()
2. platform()
3. version()
4. hostname()

import os from ‘os’;

Note: import {os} is not used instead import os is used because it is a default module

Third party module

* readline-sync
* express
* cors
* mongodb
* jwt

All these modules you can download using npm command

readline-sync: It is used to take input from the keyword

npm install readline-sync [or] npm i readline-sync

Day 2 Agenda

* Read & Write JSON files
* HTTP module
* Express Module

JSON:

Javascript Object Notation is one of the widely used datastructure to exchange the data from one application to another application, JSON data will be in key value pairs wrapped in { }

ex: {“pin”:8399, “accountNumber”:929919339}

Above JSON is a single JSON data, we can also represent multiple JSON data in an array

ex:

[  
{“date”:”20-07-2023”, “amount”:5000, “type”:”credit”},

{“date”:”21-07-2023”, “amount”:15000, “type”:”debit”},

]

Javascript Object vs JSON

Javascript object looks like below:

{id: 100, name:”Raj”, salary:35000}

JSON looks like

{“id”: 100, “name”:”Raj”, “salary”:35000}

JSON is a string data, you can’t access JSON properties directly until you convert it into the respective structure i.e., to JS object or Java object or C# object or python object & so on

If JSON is converted to Javascript object then you can access the javascript object property

let jsObj = {id: 100, name:”Raj”, salary:35000}

console.log(jsObj.id); // prints 100

let jsonObj = {“id”: 100, “name”:”Raj”, “salary”:35000}

console.log(jsonObj.id); // prints

fs - readFileSync & writeFileSync

Note: writeFileSync can only write text data to the file(txt, json and etc)

let emp = { id: 100, name: “Raj”, salary:35000 }

fs.writeFileSync(“emp.json”, emp);

// you will get an error because you can’t write Javascript object to the file

How to convert Javascript object to JSON

let jsonString = JSON.stringify(emp);

fs.writeFileSync(“emp.json”, jsonString)

Note: While writing JSON file you can’t use {flag:”a+”} as it is not a simple text data to append, since it’s a JSON data, you need to convert all the data to JSON format and write at once, it is done by keeping all the JSON in an array and writing them at once

Writing JSON data to file without erasing old data

1. Firstly you need to check if the file exists or not,
2. if file doesn’t exist write an empty array and read that array to store the new json into the array
3. if file exists then read the array present and store the new json into the array

Note: Create a module that can read & write employee data from/to file

Activity:

In the employee-util.js create reusable functions which can update employee salary based on id, delete employee based on id, read employee based on id

1. Create 3 methods as mentioned which should update the json file when you try to delete or update the employee
2. From the caller create 3 more options apart from the existing 3 options 1: Store, 2: Read -1: Exit, the options must be

1: Store, 2: Read, 3: Read Employee by Id, 4: Update Employee Salary by Id, 5: Delete Employee by Id, -1: Exit

Note: The operations must reflect to the JSON file

ex: If json file has below data   
[ {“id”:100, “name”:”Raj”, “salary”:45000}, {“id”:200, “name”:”Vijay”, “salary”:30000} ]

When you want to update salary to 50000 for on id 200, then the above data must reflect in the file as below

[ {“id”:100, “name”:”Raj”, “salary”:45000}, {“id”:200, “name”:”Vijay”, “salary”:50000} ]

http module

It is used to create servers & run applications to serve the request

import http from ‘http’;

let server = http.createServer(callbackFn);

createServer is used to create a server instance, it takes a callbackFn

callbackFn of createServer is executed when the request is sent, it has 2 arguments 1st is request & 2nd is response

function(request, response) { …. }

server.listen(port\_number, callbackFn)

listen is function is used to start the server instance from the portNumber mentioned in the 1st argument

callbackFn of listen is called when the server is started

RESTful webservice

Representational State Transfer

Reprsentational means JSON/Text/XML/HTML representation

State means data

Transfer means transferring the data

Things to keep in mind while designing webservice

1. Data structure like JSON or Text
2. URL of the webservice
3. HTTP methods to specify the operation

express:

It is a third party module which you need to download using npm command.

npm install express

To import you can use import express from ‘express’;

let app = express();

app.get(url, callbackFn)

app.post(url, callbackFn)

app.put(url, callbackFn)

app.delete(url, callbackFn)

All the callbackFn of http methods takes request & response arguments

app.listen(port, callbackFn)

callbackFn of listen doesn’t take an argument