Assignment No 3.

Q1) Explain any 2 module of Nodejs :-
→ 1) Core Module :-
a) Path module :-
The path module is one of the core mod in the nodejs. designed to handles ul path & directory. path in a platform dependent ways.

* Methods :-
- path. basename (path)
- path. dirname (path)
- path. extname (path)
- path. format (path, obj)
- path. join (... path)
- path. resolve (... path).

* Module properties :
- path. sep
- path. delemiter

b) OS Module :-
The OS module in node.js modules a lot of opearing system- releted utility methods & properties. It allows you to access information about the OS on which nodejs is ranning.

* Modules :
1) - os. arch ( )
2) os. cpus ( )
3) os. freemem ( )
4) os. hourmemo( )

5) os.plateform()
6) os. replace()
7) os. type()
8) os. homedir()
9) os. tmdir().

2) local module:
Node JS treats each JS file as a ~~sp~~ seperate module. In the Module (JS file) all the variables & function are private.

To use variable & functions of a module in another module you need to export them at the end of the JS file.

Q2) Write a program to show current date & time using user defined module in node.js.

→ dateTimeModule.JS:

```
exports. set current Date & time = function() {
        Const   currentDate - new  Date();
        return  CurrentDate - to local string();
};
```

app.js :-

```
Const dateTimeModule - require(". / dateTimeModule);

Const CurrentTime Date = require.(' current DateTimel
Console.log (" current Date & time :", currentDateTime);
```

- run proram:-

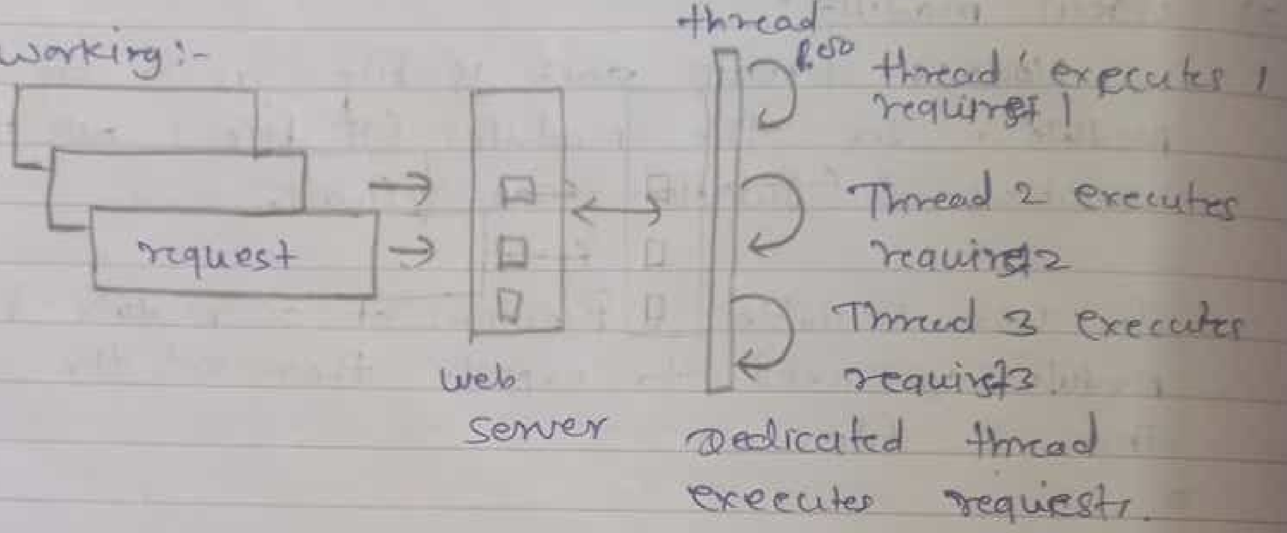
```
npm app. JS.
```

Q3) What is nodejs & explain its working & features?

Nodejs :-

Its a single threaded, open source, cross platform Javascript runtime environment, built on google open source V8 Javascript engine for bulding fast & Scalable server side & networking applications.

- working :-



request
web server
thread pool
thread 1 executes request 1
Thread 2 executes request 2
Thread 3 executes request 3
Dedicated thread executes requests.

- Feature :-
- Nodejs is single thread.
- Asynchronous by default Nodejs address blocking I/O issues by using non-blocking I/O request meaning we can continue making request while other tasks are going on

- Node Js is a event-driven.
lightweight frameworks that includes bare minimum Module other module can be include as per needs

Q4) Write a program to create a server in Node Js & display server response on web page?

→ Server JS :-

```
Const http = require (http);
```

```
const port = 3000;

Const server = Http. Createserver ((req. res)=> {
      res. writeHead (200, { contentType : text. http;
      res. write("h1) hello . world <lh1) ");
      res. end();
});

Server. listen ( port . () => {
      Console. log(" server is running -");
});
```

Run Server using Nodejs
=> Nod. server.js.

Step2:
Open a web ~~f~~ browser & navigate to 'http://localhost : 3000/'.

Q5 Write Event of Nodejs?
→ Event of Node js :-
1) EventEmmiter :-
The EventEmmiter class is a core module is nodejs that cause allow objects to emmit named event that cause function object ('listener') to be called. is provides method to emit event add listeners. & remove etc.

2) Events :-
Event are action of occurence that happen in the System.

3) Event Handlers (Listeners) :-
Event handlers also known as listers, are functions that are invoked, when a ~~part~~ particular event is emitted.

4) Event driven architecture :-
NodeJs follow an event driven application wh most of the API's are syn·chronause & non-blo This allows nodejs to handles a large number of connection efficiently.

eg

```
Const EventEmmiter= require ('event');
Const myEmitter = new EurrentEmiter();
~~Const myEventhandle occured"); 3;~~
 Const myEventhandle = () => {
        Console. log ("event occured"); 3;
myEventers.on ("myevent", myEventhandler");
myEmmiter.emit ("Myevent");
```

5) Event loop:
The event loop is key concepts in Nodejs for handling I/O operations asynchronously it continously listens for events & triggers the associated event handlers.

[illegible]

Assignment No 5.

Explain date & time filter in a angular with Example ?

In angular you can implement date & time filtering using pipes.

Make sure you have imported Date pipe form @angular/common in your components.

```
import { Date pipe} from @angular/common;
import { component } from @angular/core';
import{ Date pipe} from @angular/common;

@component ({
    selector: app-date-time-example';
    template: / date-time-example. component.html';
    style url: [:/date-time-example. component.
    css].

})

export class DateTime ExampleComponent {
    currentDate: Date = new Date ();
    currentDate: Date = new Date ("2024-03-2.
    T1T2: 00:00)}

constructor (Private datepipe: Datepipe){}

formatCurrent Date (): string {
    return this.datepipe tranform(this. currentDate.
    YYYY-MM-dd HH.mm-ss):
 }
}.
```

html :-
(P) Current Date: {{ format currentDate() }} </P>
<P> Current Date: {{ formatecurrentDate() }} </P>
=

Q 2> What is routing in agular?
→ Routing :-
Routing in agular allows you to navigate to between different component & view in your single page application (SPA) without reloading the entire page.

1) Setting up Routers :
In angular project define your routers in the app-routing-module.js file as in separate routing module.

2) Creating components :-
Create the Components referenced in the routes for example create 'home-component.js' about Component & Continous Component.

3) Navigation :-
Use angular direction such as router link to navigate between routes in your Html template. The router outlet directive will dynamically load the component asiociated with curmens source.

4) Router Parameters :-
Your can define with parameter to handle dynamic data.

Q3) Explain dependency injection in argular with example?
→ Dependency injection (DJ) :-
Dependency injection (DI) in Angular is a design pattern used to provide & manage dependencied within your application. It helps in creating looser coupled components & facilates testing reusability & maintenability.

1) Generating server :-
first a service that you want to inject into your Components. Services are typically used for sharing date or loctionality across multiple components.

2) Injection the service :-
Next iject the service into your components constructors. Angualars os system will take core os providing an infance of the service when components is created.

3) Using the service :
How you can use the service method or properties within your components.

4) Providing service in module or components :
providing service in module or components an you can also provide services of the module or components level.

5) Injecting other Dependencies:
There apart from services you can also.

inject other dependencies fin other service: HTTP client router etc.

Q4) Explain SPA in Angular :-
→ SPA (Single page application) :-
Angular is well-suited for building SPA's due to its powerful featurs like routing, data binding & dependency injection.

1) Routing setup:-
Angular's router allow you to define difference router for your application. You can figure routing in the app-routing-module's file or in seperater routing module.

2) Component structure:-
organize your application into componets each component ~~&~~ represents a part of your application UI Compements can be mested to create comple UI structure.

3) Template Binding:-
~~Use Angular data binding system syntax bind data from your components class to its template. This allow you to dynamically update the of build or changes in your application data.~~

4) Service for Business logic:-
Use Angualar service to encapsuatate ~~.~~ business logic data. access & other shared fuctionality service can be injected into compements

making them reusable ~~to~~ trustable.

5) HTTP Requests:-
Use Angualar's Http. client module to make http request to a server for testing data ~~is~~ or interacting with API's.

6) Testing:-
Write until tests & end-to-end tests to ensure the reliability & correctness of your applications.

Q5.) Explain Typescript or angular with example?
→

```
import { components} from '@angular/core';
import { httpclient} from '@ angular/ Common / http'}
import { observable} from 'rxjs';

@component ((
    selector: app-root";
    templateurl:: /app. Component. html;
    Styleurl: [:/ app. component. css :]
 })
export class Appcomponents{
    users $: observable <any>;

    constructor ( private http. http client ) {
        this .users $ = this. http. get (' http://jsonplace
        holder. typicode. com/ users ");
    }
}
```

we import necessary module like 'component', 'httpClient' & 'observable' from Angular. core modules.

We define Typescript class App. Component ~~for~~. represent a component in Angular within app component class we declare a variable 'users $' of type 'observable <any>' to hold data retained from API synchronosly.

Inside the constructor we make HTTP GET request using 'httpClient' service to fetch the data.

[illegible]