■ Event of the Day Date:...../...../.....

Node.js

⇒ Node.js is an open source, cross-platform Javascript runtime environment for running web applications outside the client's browser.

⇒ It is used for creating server side web apps.

- Advantages:

⇒ It is fast.

⇒ All APIs of node.js library are asynchronous, i.e they never wait for API to return data.

⇒ Better synchronisation of code b/w client & server due to same code base.

⇒ Cross Platform.

Parts of Node.js:

1) Modules
2) Console
3) Cluster
4) Global
5) Error Handling
6) Streaming
7) Buffer
8) Domain
9) DNS
10) Debugger

■ Event of the Day

Date:...../...../.....

1) Modules:

⇒ They are used to include a set of functions.

⇒ To include a module in Node.js application, use require(') containing the name of the module.

Eg. var http = require('http');

⇒ See core modules & their description.

2) Console:

⇒ It is a module that provides a way to debug

⇒ It prints messages to stdout & stderr. ∴ Console.log("Hi");

3) Cluster:

⇒ Cluster is a module that allows multithreading by creating child processes that share the same server port & run simultaneously.

Eg.
```
var cluster = require('cluster');
if(cluster.isWorker){
    console.log('child thread'); }
else console.log('Parent');
```

■ Event of the Day

Date:...../...../.....

4) Global :

⇒ Global objects are functions, modules, strings etc.

⇒ eg. _dirname, _filename, exports, modules, require etc.

∴ See ss.

5) Error Handling: 1) Std. Javascript Error 3) User Specific error

2) System Errors 4) Assertion

6) Streaming :

⇒ They are the objects that let you R/W data continously

⇒ 4 types of streams : 1) Readable 3) Duplex

2) Writable 4) Transform

7) Buffer :

⇒ It is a module that allows handling streams that contain only binary data.

⇒ An empty buffer of length '10' can be created by this method :

```
var buf = Buffer.alloc(10);
```

■ Event of the Day Date:...../...../.....

8) Domain :

⇒ Domain module intercepts errors that remain unhandled

⇒ 2 methods:

1) Internal Binding : Error emitter executes code inside the run method.

2) Ext. binding : Error emitter is explicitly added to a domain via it's add method.

9) DNS:

⇒ It is used to connect to DNS server and perform name resolution by using : dns.resolve()

⇒ It is used for performing name resolution w/o a N/w communication by : dns.lookup().

10) Debugger :

⇒ Debugger can be used in the terminal by :

$node inspect myscript.js

↑ Keyword ↑ JS file name

■ Event of the Day

Date: ...../...../.....

Express:

⇒ It is a framework that provides a wide set of features to develop both web & mobile app.

⇒ app.get('/', function(req, res) {})

∴ The request object represents HTTP request.
The response object represents HTTP response that an express app sends when it gets HTTP req.

How to run a node program:

⇒
```
var http = require('http');
http.createServer(function(req, res){
    res.write('Hello');
    res.end();
}).listen(8080);
```
Save this

⇒ On cmd, navigate to folder, & write
node demo1.js → File name

⇒ Open browser & type: localhost:8080

■ Event of the Day Date:...../...../.....

Architecture:

⇒ Single-threaded event loop allows to handle multiple concurrent clients.

1) Clients send req. to web server

2) Node.js receives the req. & places them in event queue.

3) Node.js internally maintains a thread pool.

4) Event loop receives req. & processes them.

5) Thread Pool interacts with external resources like Database.

Node Package Manager (npm):

⇒ Responsible for providing all the packages & modules of node.js.

⇒ Provides cmd line utility to install node packages.

■ Event of the Day Date:...../...../.....

Express Features:

⇒ It can be used to design single page, multi-page, hybrid web apps.

⇒ Allows to setup middlewares to respond to HTTP req.

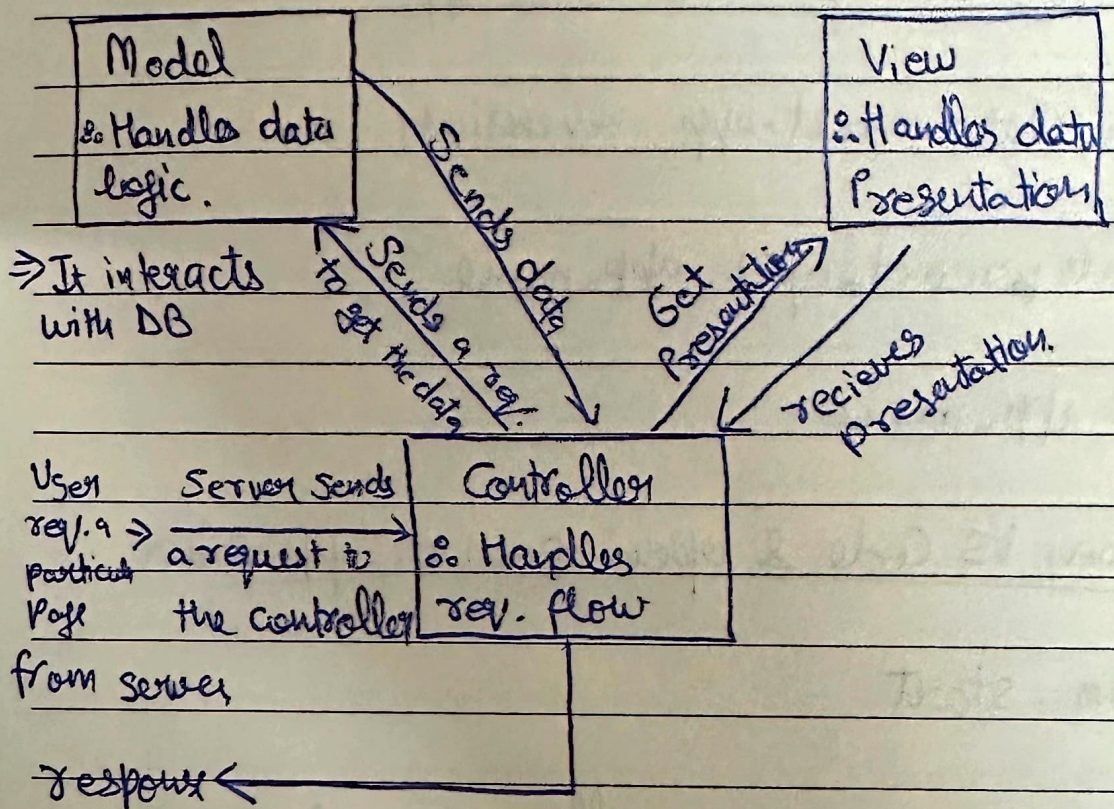
⇒ It is ~~used~~ defines a routing table which is used to perform different actions based on HTTP method & URL.

Express Code to print Hello World:

```
Var express = require ('express');
Var app = express();

app.get (function (req, res) {
    res.send ('Hello World');
})

Var server = app.listen (8081, function() {
    Var host = server.address().address.
    Var port = server.address().port

    Console.log ("Example app listening at http://%s:%s", host, port)
})
```

■ Event of the Day Date: ..../..../.....

MVC Pattern: Goal of this pattern is to split a large application into specific sections that all have their own purpose.



2) **GET**: The operation reads info from a record in the DB
PUT: The operation changes a record's info in the DB
POST: The operation creates a new record in the DB.

avantor

Express.js is a minimalist web application framework for Node.js. It provides a robust set of features to build web and mobile applications. Express simplifies the process of handling HTTP requests, routing, middleware integration, and more. It's widely used in building APIs, single-page applications, and full-stack web applications due to its flexibility and scalability.

- **REST API:** REST (Representational State Transfer) APIs are a set of architectural principles for building web services. They use standard HTTP methods like GET, POST, PUT, and DELETE to perform CRUD operations on resources, typically using JSON for data exchange.

- **HTTP Code:** HTTP status codes are standardized responses from a server to a client's request. They indicate the success or failure of the request and provide information about the outcome, such as 200 for success, 404 for not found, or 500 for server errors.

- **Streams:** Streams in Node.js provide an efficient way to handle data flow, enabling you to read or write data in chunks rather than loading it all into memory at once. They're particularly useful for processing large files or network data.

- **Express Middleware:** Middleware functions in Express.js are functions that have access to the request and response objects in an HTTP request-response cycle. They can modify the request or response, execute code, or terminate the request-response cycle.

- **JWT:** JWT (JSON Web Token) is a compact, URL-safe means of representing claims securely between two parties. It's commonly used for authentication and authorization in web applications, allowing users to securely access resources after authentication.

- **App Security:** App security involves implementing measures to protect web applications from various threats, such as injection attacks, cross-site scripting (XSS), cross-site request forgery (CSRF), and other vulnerabilities.

- **SOLID:** SOLID is a set of principles for writing maintainable, scalable, and flexible software. It stands for Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, and Dependency Inversion, guiding developers to write clean and modular code.

- **Design Pattern:** Design patterns are reusable solutions to common problems encountered in software design. They provide templates for solving recurring design challenges, promoting code reusability, maintainability, and scalability.

- **ORM and ODM:** ORM (Object-Relational Mapping) and ODM (Object-Document Mapping) are techniques used to abstract database interactions by mapping database objects to application objects. ORM is for relational databases like SQL, while ODM is for document databases like MongoDB.

- **Rate Limiter:** Rate limiting is a technique used to control the rate of requests sent or received by a server or API. It helps prevent abuse, improve security, and ensure fair usage by limiting the number of requests a client can make within a specified time period.

- **Logger:** Logging in Node.js involves recording information about application events, errors, and behaviors to aid in debugging, monitoring, and troubleshooting. Loggers provide various levels of logging, such as info, debug, warn, and error, and can output to console, files, or other destinations.