IT3501-FULL STACK WEB DEVELOPMENT

UNIT-I BASICS OF FULL STACK

QUESTION BANK

1) What is web development framework?

Ans

- A web development framework is a collection of tools, libraries and technologies that a used to create web applications and websites.
- It provides a basic structure and set of conventions for developers to build web applications quickly and efficiently

2)Why do we use web framework? Give any two reasons.

Ans:

1. Saves time: The most striking feature of web framework is that it saves time and energy in developing any app because the developer doesn't need to worry about session handling. error handling and authentication logic. These functions are taken care of by web frameworks.

2. Flexibility and customizable : Add-ons, themes, plugins, widgets enable rapid customization on the web application. This brings out a lot of flexibility in code development

3)What are the components of web development framework?

Ans.:

The components of web development framework are-

(1) User

(2) Browser

(3) Web Server

(4) Backend services

4)Give the importance of browser

ans.:

The browser plays three roles in web framework

It provides communication to and from the web server.

It interprets the data from the server and displays the visual output.

The browser handles user interaction through the keyboard, mouse, touchscreen or other input device and takes the appropriate action.

5)What is GET and POST request ?

Ans :

i) GET: The GET request is typically used to retrieve data from the server. This data can be inthe html files or images.

ii) POST: The POST request is used for sending the data to the server. For example - Creditcard details while performing online shopping

6)What is web server?

Ans.:

Web server is a special type of server to which the web browser submits the request of a web page which is desired by the client.

7)Enlist the functions of web server

Ans.: Various functions of web server are -

1. The web servers accept the requests from the web browsers.
2. The user request is processed by the web server.
3. The web servers respond to the users by providing the services which they demand for over the web browsers
4. The web servers serve the web based applications.

8)What do you mean by backend services

Ans.:

1) Backend services are the services that run behind the web server. These services provide th data to the web server which is requested by the web browser.

2) The most commonly used backend service is a database that stores the information.

9)What is MVC?

Ans.:

1. Model: This part of the architecture is responsible for managing the application data. The module responds to the request made from view. The model gives instructions to controller to update when the response is made.
2. View: This part takes care of the presentation of data. The data is presented in des format with the help of view. This is a script based system using JSP, ASP, PHP and so on.

3. Controller: The controller receives input, validates it and then performs business operation that modify the state of the data model. The controller basically responds to user required and performs interaction with the model.

10)What is full stack ?

Ans: Full stack development is development of both front end and back end of any web application The full stack consists of - 1. MongoDB 2. Express 3. AngularJS 4. Node.js 5. React.js

11.Difference between frontend and back end technologies?

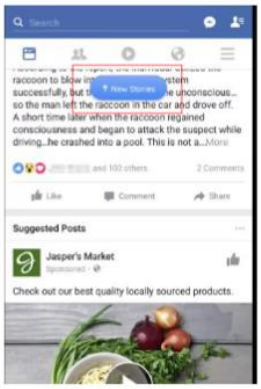
Here is a difference table for Front end Vs. Backend Web Development Frameworks:

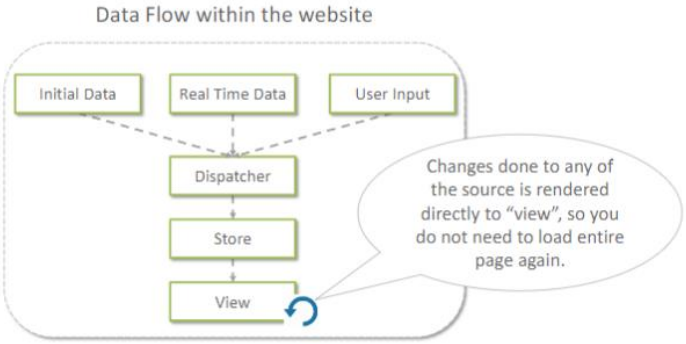
Front End Frameworks	Backend Frameworks
The front end is the part of the web app that interacts with the end-user.	The backend of an app refers to the background functioning of the web application.
It offers pre-written code snippets, integrable elements, reusable templates, and manage user interactions.	It offers features such as user authorization, privacy encryptions, database manipulation, reusable components for connecting the front end to the backend.
The key features include reusable templates, UI/UX designing, SEO optimization, and enhancing the performance and scalability of the app.	The key features include server handling, designing site architecture, database management, security, and URL routing.
HTML, CSS, JavaScript, JQuery, etc., are a few prevalent front-end languages.	Python, JavaScript, PHP, Ruby, .NET, etc., are a few prominent backend languages.
React, Vue, Ember, Bootstrap, and Angular are the most famous front-end frameworks.	Django, Ruby On Rails, Spring, Express, ASP.NET Core are the most famous backend frameworks.

12.Why Reactjs is used in Facebook?

Facebook: Solution

Facebook after implementation of React became more user friendly. When new data is added, a "New Stories" notification appears. Clicking it, will automatically refresh the newsfeed section





React uses ***virtual DOM*** which optimizes the memory consumption and results to faster load of web pages.

13.Comapre Angular,Node js,React

Angular Js	Node Js	React Js
Uses MVVC pattern	Event driven architecture	Uses Controller View pattern
Designed for single page application development	For highly concurrent server side design	Best for building scalable network applications
Open source java script framework.	Run time environment	Open source java script library.
Used for front end tasks	It's a Javascript run time	Can create view in client side and server side.

14. What Is Express JS?

Express is a **node js web application *framework*** that provides **broad features for building web and mobile applications**.

It is used to build **a single page, multipage, and hybrid web application**.It's a layer built on the top of the Node js that helps manage servers and routes.

15.Difference Between SQL and NoSQL

Entity	SQL Databases	NoSQL Databases
Type	One Type – SQL (with slight variations)	Multiple Types – Document, Key-Value, Tabular, Graph
Developed In	1970	2000
Examples	Oracle, MSSQL, DB2 etc.	MongoDB, Cassandra, HBase, Neo4J
Schemas	Fixed	Dynamic
Scaling	Vertical	Horizontal
Dev Model	Mix	Open Source
Properties Followed	ACID	BASE

Part B Questions

1.Explain the Basics of Web Development Framework?

Define WebDevelopment Framework

Need of WebDevelopment Framework

Reasons for Web Development Frameworks used for App Development Process

- Higher Performance
- Reduction in Errors
- Faster Development
- Better Reliability and Security

Types of Web Development Frameworks

- Front End Web Development Framework
- Backend Web Development Framework

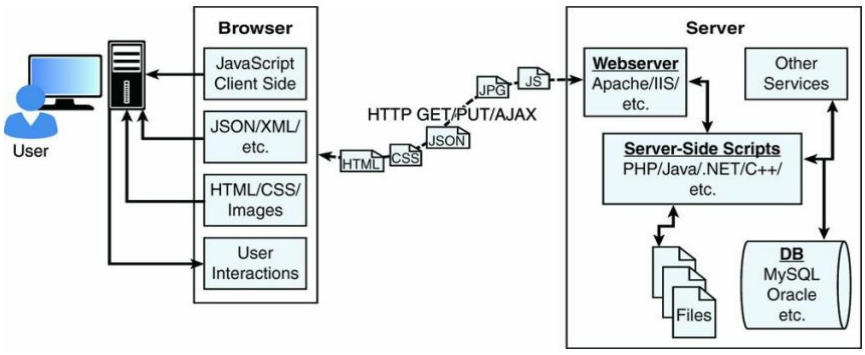


Figure 1.1 Diagram showing the components of a basic website/web application

2.Explain the components of WebDevelopment Framework?

The components of the WebDevelopment framework

(i)User (ii)Browser (iii)Web Server(iv)Backend services

User

- Definition

Browser

- Roles of the Browser
- Browser to Webserver Communication
- Three main types of requests to the server
- Rendering the Browser View
- Architecture of a Browser
- Working of Browser

WebServer

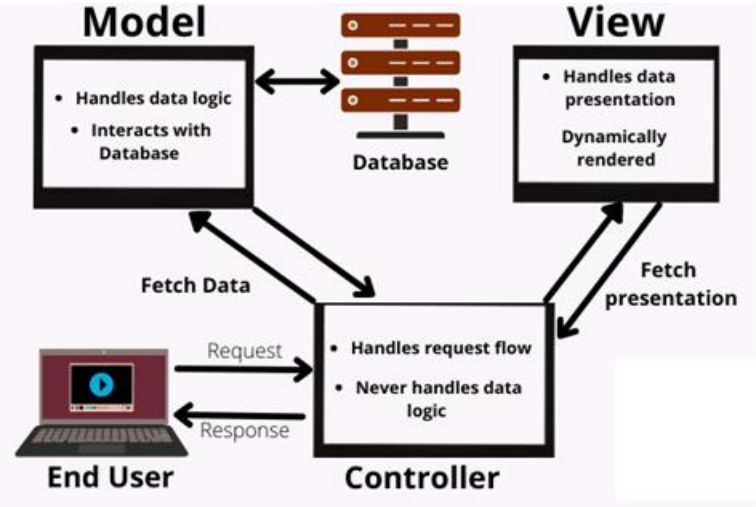
Backend Services

3.Explain MVC Architecture in detail?

Definition

Features of MVC

Components of MVC



Example

Advantages of MVC

Disadvantages of MVC

4. Explain the different stacks in webDevelopment?

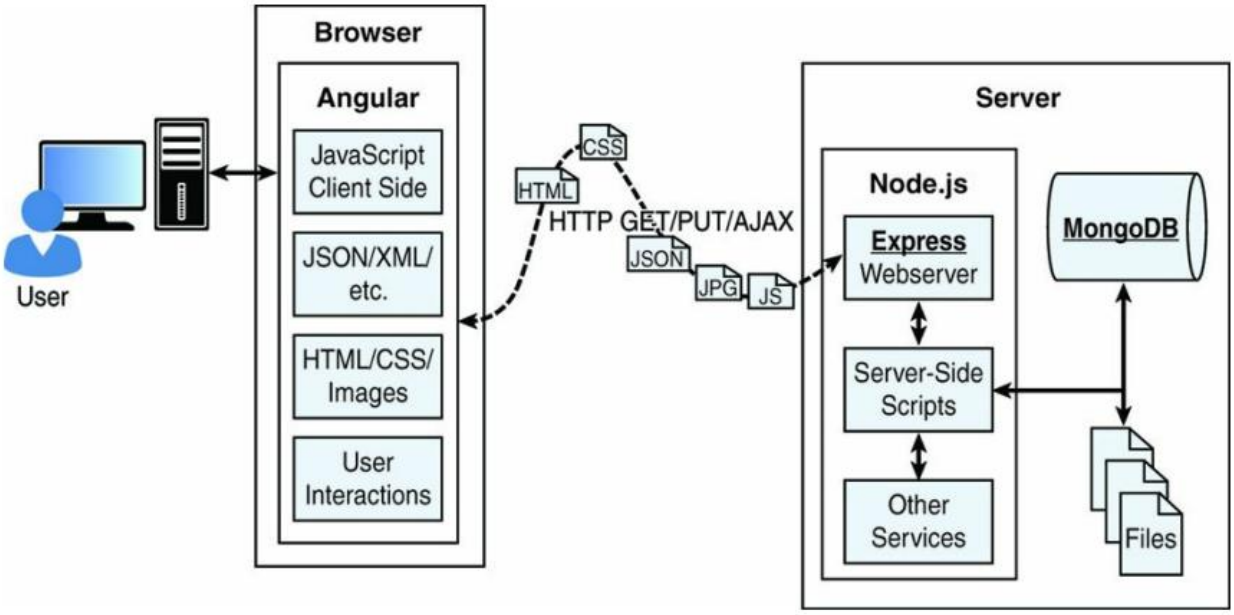


Figure 1.2 Basic diagram showing where Node.js, Express, MongoDB, and Angular fit in the web paradigm

- Nodejs
- Reasons of Nodejs
- MongoDB
- Express
- Features of Express
- Angular

5.Explain the role of

(i)Express (ii) Angular (iii) Node (iv) Mongo DB (v) React

Express

- Definition
- Need Of Express JS
- Features of ExpressJS
- Advantages and disadvantages of Express JS
- Simple Program in ExpressJS

Angular

- Definition
- Need Of Angular

- Features of Angular
- Advantages and disadvantages of Angular
- Simple Program in ExpressJs

MongoDB

- Define
- Difference between NOSQL And MongoDB
- Features of
- Architectures of MongoDB

React

- Definition
- Need of React
- Component Based
- No Templates
- Isomorphic

IT3501-FULL STACK WEB DEVELOPMENT

UNIT =II NODE JS

PART A QUESTIONS

1.What is Node.js ? Give its uses.

Ans.: Node JS is an open source technology for server. Using Node.js we can run JavaScript on server.

It can perform various tasks such as -

1) It can create, open, read, delete, write and close files on the server.

2) It can collect form data.

3) It can also add, delete, modify data in databases.

4) It generate dynamic web pages.

2 What is NPM?

Ans: The NPM stands for Node Package Manager. NPM consists of two main parts:

1) A CLI (command-line interface) tool for publishing and downloading packages,

2) An online repository that hosts JavaScript packages

3.Explain the use of 'cluster module in Node.js

Ans: The cluster module helps in creating a child process. Both the main and child can run simultaneously and share the same server port

4.Explain the console.log statement in Node.js with suitable example

Ans: console.log(data][,]): It is used for printing to stdout with newline. This function take maltiple arguments similar to a printf statement in C

For example

```
var emp_name= 'Ankita'

emp_dept {

dept: 'Sales', dept2: 'Accounts'
```

};

console.log (Name %s+\n+ Departments:%f,emp_name, emp depts);

5.What is EventEmitter ?

Ans: The emitter objects performs following tasks-

1) It emits named events

2) Registed and unregisted listener functions.

Basically EventEmitter is a class which can be used to raise and handle custom even.

6.What is the purpose of addListener method?

Ans: Adds a listener to the end of the listeners array for the specified event. No checks made to see if the listener has already been added

7.Explain the setMaxListener and getMaxListener methods of Node.js

Ans:

1) setMaxListeners(int): This method sets the number of listeners to the particular event.

default, a maximum of 10 listeners can be registered for any single event.

2) getMaxListeners(): It will return the max listeners value set by setMaxListeners()

8.Explain the setTimeout function in Node.js with suitable

Ans: This function is used to set one time callback after delaying by some milliseconds.

Syntax

Timeout(function.delay in milliseconds)

Example code

Console.log(Task1')

Timeout(function() (console.log("Taks2: Executing After 4 seconds")),4000);

console.log("Task3)

9.How callback functions are handled in Node.js ?

Ans: Callbacks is a function which is usually passed as an argument to another function and it is usually invoked after some kind of event is completed.

```
// A function that simulates an asynchronous operation
function simulateAsyncOperation(callback) {
  setTimeout(() => {    const randomValue = Math.random();
    if (randomValue < 0.5) {
      // Simulate an error
      callback(new Error('Operation failed'), null);
    } else {
      // Simulate success
      callback(null, 'Operation completed successfully');
    }
  }, 1000); // Simulate a 1-second delay
}
// Calling the function with a callback
simulateAsyncOperation((err, result) => {  if (err) {
    console.error('Error:', err.message);
  } else {    console.log('Result:', result);
  }});
```

10. What is pipe operation in Node.js.

Ans: The pipe operation is a kind of operation in which output of one stream acts as an input of another stream. There is no limit on pipe operation that means, all the output of one stream can be fed as input to another stream.

pipeExample.js

```
Var fs =require("fs")

Var rs=fs.createRoadStream(input.txt');

rs.setEncoding(8)

var ws=fs.createWriteStream('output.txt');

rs.setEncoding("utf8");

rs.pipe(ws)

console.log("Data is transfered from input.txt' to 'output.txt" ");
```

11.What is QueryString?

Ans: The query string specifies the parameters of the data that is being queried from a website's database. Each query string is made up of a parameter and a value joined by the equals (-) sign. In case of multiple parameters, query strings are joined using the ampersand (&) sign. The parameter can be a number, string, encrypted value or any other form of data on the database.

12.Explain the use of ClientRequest object.

Ans. The ClientRequest object is used to monitor and handle the response from the server. For implementing the ClientRequest object the call to the http.request() is made.

The syntax to this method is as follows-

```
http.request(options.callback)
```

13.What is event loop?

The **event loop** allows Node.js to perform **non-blocking I/O operations** despite the fact that JavaScript is single-threaded.

It is done by assigning operations to the operating system whenever and wherever possible.

14.What is node package Module?

Definition of Node Packaged Modules

A **Node Packaged Module** is a **packaged library** that can **easily be shared, reused, and installed in different projects**.

15.What is Package.json?

Package.json

It is the ***manifest file of any Node.js project***,which contains **all the metadata of a project.**

16.What is promise?

To manage asynchronous actions in JavaScript, promises are used. It is an assurance that something will be done.

The promise is used to keep track of whether the asynchronous event has been executed or not and determines what happens after the event has occurred.

A Promise has four states:

fulfilled: Action related to the promise succeeded

rejected: Action related to the promise failed

pending: Promise is still pending i.e. not fulfilled or rejected yet

settled: Promise has fulfilled or rejected

17. **Difference Between SetTimeout and SetTimeInterval**

SetTimeout	**SetTimeInterval**
Delay a operation	Delayed operation in a loop
This is a **time event function used to call another function after certain time** period but it executes **the function only once.**	Here the setInterval function calls a function after a specific time period but the execution occurs continuously according to the specified time interval. Here the time interval work like a time gap after which the function has to call the another function every time.

Syntax	
setTimeout(< Function or code >, < delay in ms >, [argument 1], [argument 2], …)	setInterval(< Function or code >, < delay in ms >, [argument 1], [argument 2], …)
To stop the execution of the command when the respective command is already executed, we invoke the stopping function known as **clearTimeout,** and the variable returned by the setTimeout needs to be passed as an argument.	To stop the execution of the command when the respective command is already executed, we invoke the stopping function known as **clearInterval,** and the variable returned by the **setInterval** needs to be passed as an argument.

18.What is JSON?

JSON

- JSON (JavaScript Object Notation) is a **lightweight data *interchange*** format that is easy for **humans to read and write, and easy for machines to parse and generate**.
- It is commonly used for structuring data in a format that is **both human-readable and machine-readable.**
- JSON is often used for transmitting data between **a server and a web application**, as well as for configuration files and other data storage purposes.

19.What is buffered data?

Buffered data is made up of a **series of octets[sequence of 8 bits] in big endian or little endian format**.

Need of Buffered Data

That means they **take up considerably less space than textual data**. Therefore,

- Node.js provides the **Buffer module that gives you the functionality to create,**

read, write, and manipulate binary data in a buffer structure.

20.What are the compression methods used in Zlib?

The compression methods **supported by Zlib are**

- **gzip/gunzip:** Standard gzip compression
- **deflate/inflate:** Standard deflate compression algorithm based on Huffman coding
- **deflateRaw/inflateRaw:** Deflate compression algorithm on a raw buffer

Part B Questions

1.List the builtin Modules of Nodejs?

- **Assertion testing:** Allows you to test functionality within your code.
- **Buffer:** Enables interaction with TCP streams and file system operations.
- **C/C++ add-ons:** Allows for C or C++ code to be used just like any other Node.js module.
- **Child processes:** Allows you to create child processes.
- **Cluster:** Enables the use of multicore systems.
- **Command line options:** Gives you Node.js commands to use from a terminal.
- **Console:** Gives the user a debugging console.
- **Crypto:** Allows for the creation of custom encryption.
- **Debugger:** Allows debugging of a Node.js file.
- **DNS:** Allows connections to DNS servers.
- **Errors:** Allows for the handling of errors.
- **Events:** Enables the handling of asynchronous events.

- **File system:** Allows for file I/O with both synchronous and asynchronous methods.
- **Globals:** Makes frequently used modules available without having to include them first.
- **HTTP:** Enables support for many HTTP features.
- **HTTPS:** Enables HTTP over the TLS/SSL.
- **Modules:** Provides the module loading system for Node.js.
- **Net:** Allows the creation of servers and clients.
- **Path:** Enables access to file and directory paths.
- **Process:** Provides information and allows control over the current Node.js process
- **Query strings:** Allows for parsing and formatting URL queries.
- **Readline:** Enables an interface to read from a data stream.
- **REPL:** Allows developers to create a command shell.
- **Stream:** Provides an API to build objects with the stream interface.
- **String decoder:** Provides an API to decode buffer objects into strings.
- **Timers:** Allows for scheduling functions to be called in the future.
- **TLS/SSL:** Implements TLS and SSL protocols.
- **URL:** Enables URL resolution and parsing.
- **Utilities:** Provides support for various apps and modules.
- **V8:** Exposes APIs for the Node.js version of V8.
- **VM:** Allows for a V8 virtual machine to run and compile code.
- **ZLIB:** Enables compression using Gzip and Deflate/Inflate.

2.How NodeJs can be Installed ?

- Looking at the Node.js Install Location
- Verify Node.js Executables
- Selecting a Node.js IDE

3. What Are Node Packaged Modules?

A Node Packaged Module is a **packaged library that can easily be shared, reused,**

and installed in different projects. Many different modules are available for a variety of purposes.

For example, the Mongoose module provides an ODM (Operational Data Model) for MongoDB, Express extends Node's HTTP capabilities, and so on. Node.js modules are created by various third-party organizations to provide the needed features that Node.js lacks out of the box.

This community of contributors is active in adding and updating modules.

Node Packaged Modules include a **package.json** file that defines the packages.

The package.json file includes informational metadata, such as the name, version author, and contributors, as well as control metadata, such as dependencies and other requirements that the Node Package Manager uses when performing actions such as installation and publishing.

Directives used in the package.json files

Table 3.2 **Directives used in the package.json file**

Directive	Description	Example
name	Unique name of package.	"name": "camelot"
preferGlobal	Indicates this module prefers to be installed globally.	"preferGlobal": true
version	Version of the module.	"version": 0.0.1
author	Author of the project.	"author": "arthur@???.com"
description	Textual description of module.	"description": "a silly place"
contributors	Additional contributors to the module.	"contributors": [{ "name": "gwen", "email": "gwen@???.com"}]
bin	Binary to be installed globally with project.	"bin: { "excalibur": "./bin/excalibur"}
scripts	Specifies parameters that execute console apps	"scripts" { "start": "node ./bin/

4. How Packages are installed ,published and removed by using Node package manager?

- The Node Package Manager you have already seen is a command-line utility.
- It allows you to find, install, remove, publish, and do everything else related to Node Package Modules.
- The Node Package Manager provides the link between the Node Package Registry and your development environment.

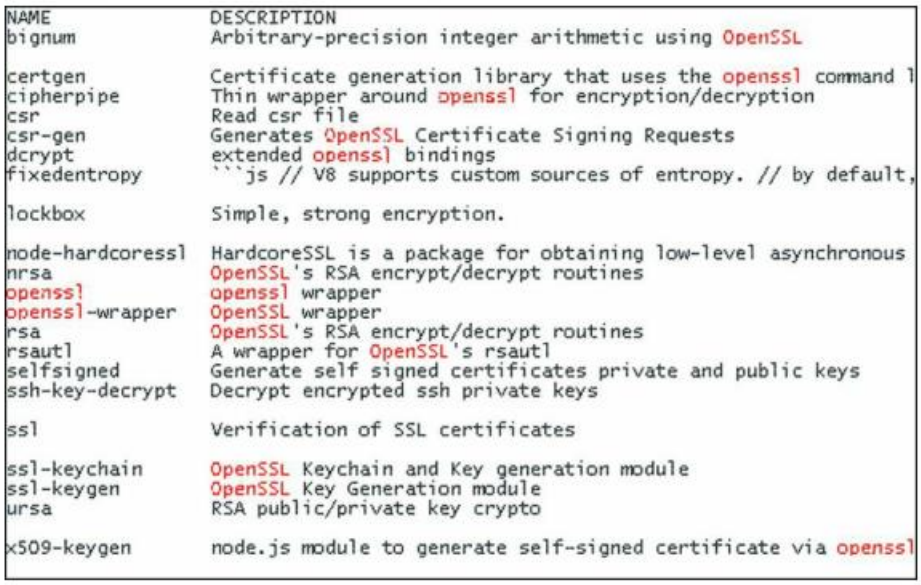
Table 3.1 **npm command-line options (with `express` as the package, where appropriate)**

Option	Description	Example
`search`	Finds module packages in the repository	`npm search express`

`install`	Installs a package either using a `package.json` file, from the repository, or a local location	`npm install`
		`npm install express`
		`npm install express@0.1.1`
		`npm install ../tModule.tgz`

`install -g`	Installs a package globally	`npm install express -g`
`remove`	Removes a module	`npm remove express`
`pack`	Packages the module defined by the `package.json` file into a .tgz file	`npm pack`
`view`	Displays module details	`npm view express`
`publish`	Publishes the module defined by a `package.json` file to the registry	`npm publish`
`unpublish`	Unpublishes a module you have published	`npm unpublish myModule`

Searching for Node Package Modules



NAME	DESCRIPTION
bignum	Arbitrary-precision integer arithmetic using OpenSSL
certgen	Certificate generation library that uses the openssl command l
cipherpipe	Thin wrapper around openssl for encryption/decryption
csr	Read csr file
csr-gen	Generates OpenSSL Certificate Signing Requests
dcrypt	extended openssl bindings
fixedentropy	```js // V8 supports custom sources of entropy. // by default,
lockbox	Simple, strong encryption.
node-hardcoressl	HardcoreSSL is a package for obtaining low-level asynchronous
nrsa	OpenSSL's RSA encrypt/decrypt routines
openssl	openssl wrapper
openssl-wrapper	OpenSSL wrapper
rsa	OpenSSL's RSA encrypt/decrypt routines
rsautl	A wrapper for OpenSSL's rsautl
selfsigned	Generate self signed certificates private and public keys
ssh-key-decrypt	Decrypt encrypted ssh private keys
ssl	Verification of SSL certificates
ssl-keychain	OpenSSL Keychain and Key generation module
ssl-keygen	OpenSSL Key Generation module
ursa	RSA public/private key crypto
x509-keygen	node.js module to generate self-signed certificate via openssl

Figure 3.2 Searching for Node.js modules from the command prompt

Installing Node Packaged Modules

- To install a Node module, use the **npm install <module_name>** command.
- This downloads the Node module to your development environment and places it into the node_modules folder where the install command is run.
- For example, the following **command installs the express module**:
- **npm install express**
- The output of the npm install command displays the dependency hierarchy installed with the module.

For example, the following code block shows part of the output from installing the express module.

Output of the npm install

```
C:\express\example
`-- express@4.14.0

+-- accepts@1.3.3
| +-- mime-types@2.1.11
| | `-- mime-db@1.23.0
| `-- negotiator@0.6.1
+-- array-flatten@1.1.1
+-- content-disposition@0.5.1
+-- content-type@1.0.2
+-- cookie@0.3.1
+-- cookie-signature@1.0.6
+-- debug@2.2.0
| `-- ms@0.7.1 ...
```

5.Explain how Event Model works in Nodejs?

- ❖ **Define EventModel**
- ❖ **Comparing Event Callbacks and Threaded Models**
- ❖ **Blocking I/O in Node.js**
- ❖ **Add Work to Event Queue**

Threaded Model

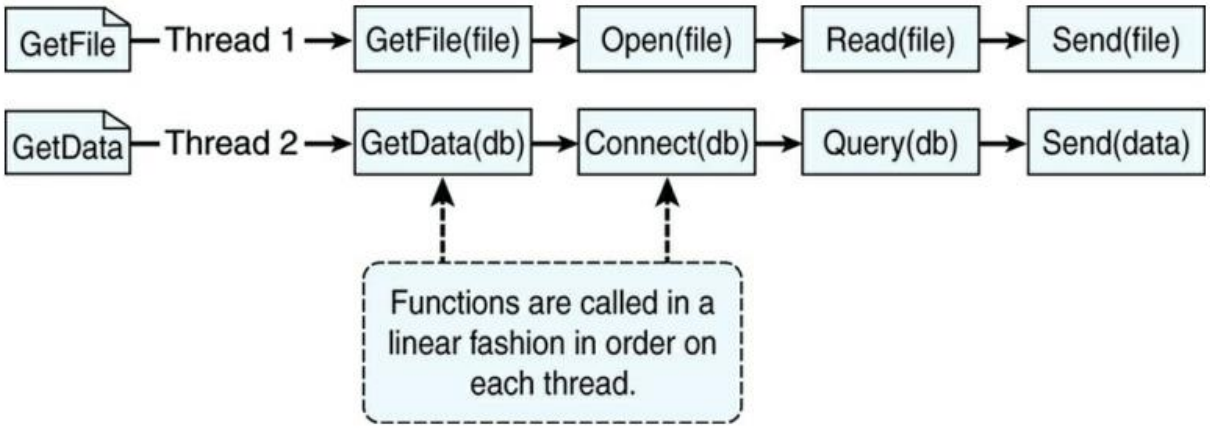


Figure 4.1 Processing two requests on individual threads using the threaded model

CallBack Model

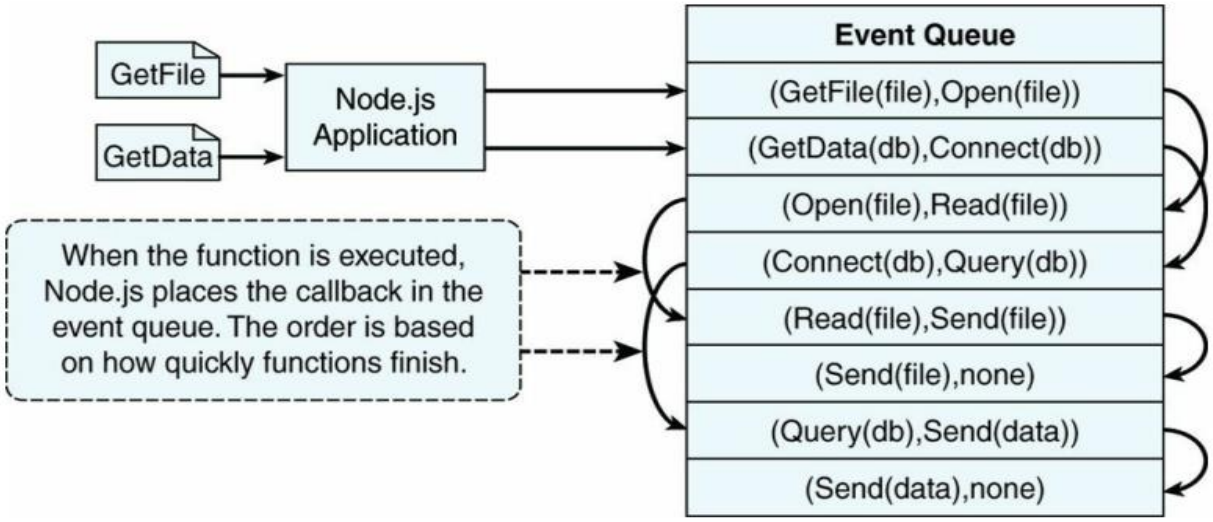
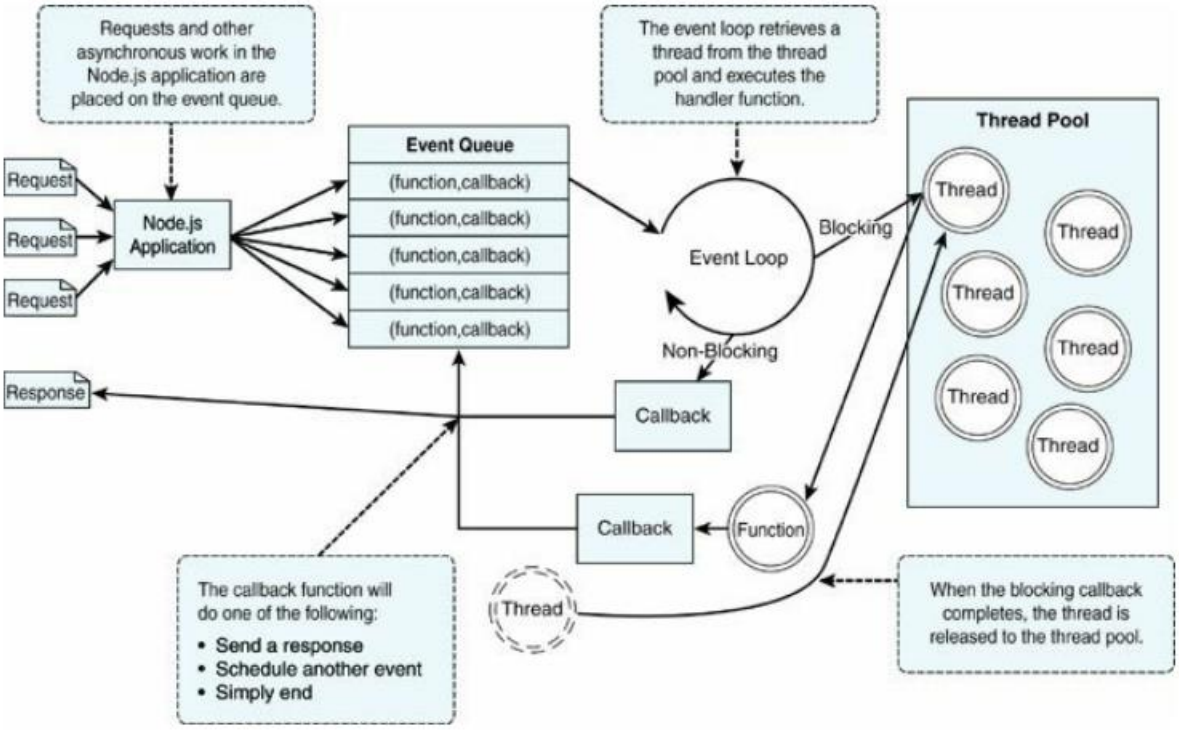


Figure 4.2 Processing two requests on a single event-driven thread using the Node.js event model



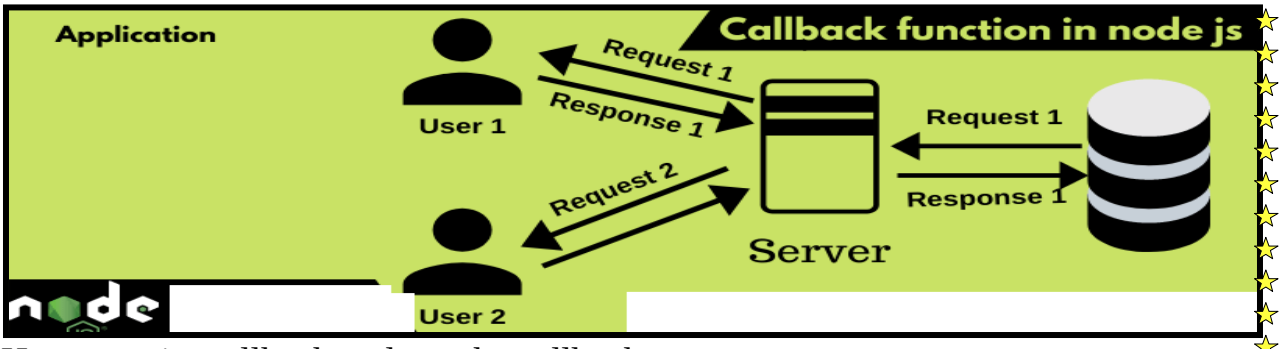
6.Why timers are used in Nodejs?

- ❖ Need of timer

- ○setImmediate().
- ○setInterval()
- ○setTimeout():
- ○nextTick

7.What is callback ?How it works in Nodejs?

Definition



- ❖ How to write callback and use the callback
- ❖ Passing Additional Parameters to Callbacks

- ❖ Implementing Closure in Callbacks
    - Definition of Closure
- ❖ Callback Chaining

8.How Data Handling is done in Nodejs?

- Understanding Buffered Data
- Creating Buffers
- Writing to Buffers
- Reading from Buffers
- Determining Buffer Length
- Copying Buffers
- Slicing Buffers
- Concatenating Buffers

9.How File Handling is done in Nodejs?

- Define Data Stream
- Readable Streams
- Writable Streams
- Duplex Streams
- Transform Streams
- Piping Readable Streams to Writable Streams

10.How Compressing and Decompressing data in Nodejs?

- Define Zlib
- Compressing and Decompressing Data with Zlib
- Compressing and Decompressing Buffers
- Compressing/Decompressing Streams

11.Explain Implementation of HTTP Services in Nodejs?

- Processing URLs
- Resolving the URL Components
- Processing Query Strings and Form Parameters
- Understanding Request, Response, and Server Objects
- The http.IncomingMessage Object

12.NodeJs Programs (All Lab Programs)

UNIT III MONGO DB
QUESTION BANK
Part-A

1.What is NOSQL?

Ans:

- **NoSQL** stands for **not only SQL**,
- It is nontabular database system that store da differently than relational tables.
- There are various types of NOSQL databases such document, key-value, wide column and graph.
- Using NOSQL we can maintain flexible schemas and these schemas can be scaled easily with large amount of data

2 List the features of NoSQL.

Ans:

1) The NoSQL does not follow any relational model.
2) It is either schema free or have relaxed schema. That means it does not require specific definition of schema.
3) Multiple NoSQL databases can be executed in distributed fashion.
4) It can process both unstructured and semi-structured data.
5) The NoSQL have higher scalability.

3. Why NoSQL?

The concept of **NoSQL (Not Only SQL)** consists of technologies that ***provide storage and retrieval without the tightly constrained models*** of traditional SQL relational databases.

The motivation behind NoSQL is mainly ***simplified designs, horizontal scaling, and finer control of the availability of data***

4.List the features of MongoDB

Ans

1) It is a schema-less, document based database system.
2) It provides high performance data persistence.
3) It supports multiple storage engines.
4)It has a rich query language support.
5) MongoDB provides high availability and redundancy with the help of replication.That means it creates multiple copies of the data and sends these copies

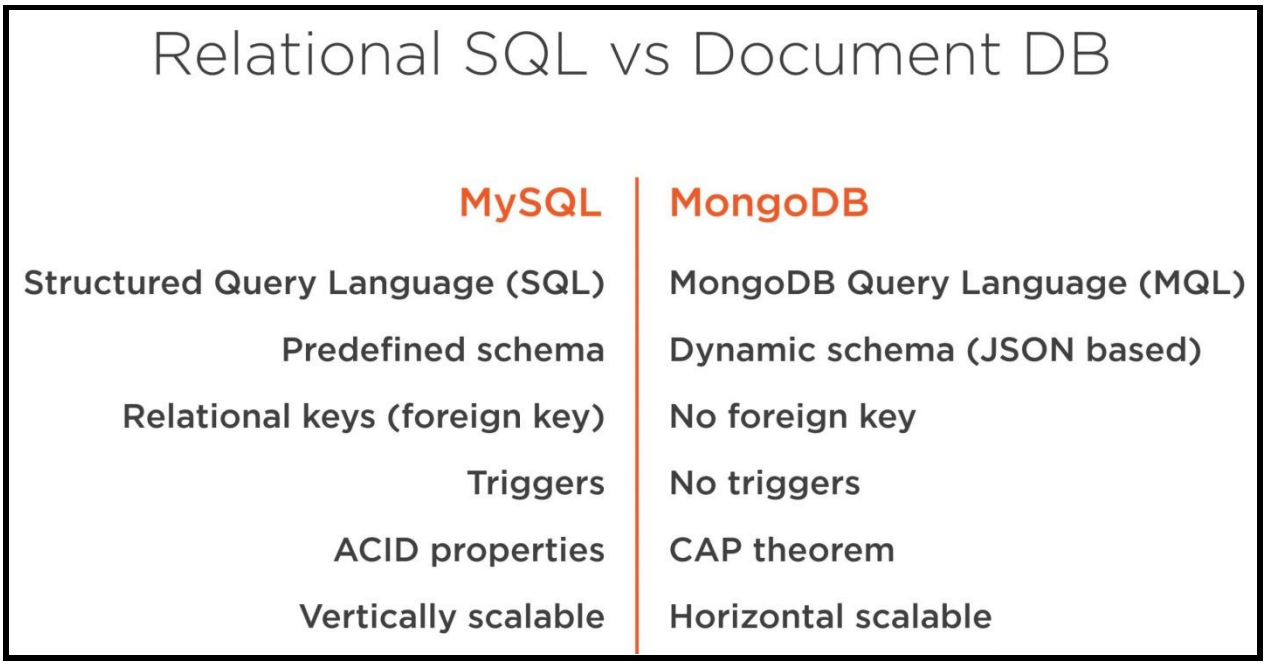
to a different server o that if one server fails, then the data is retrieved from another server.

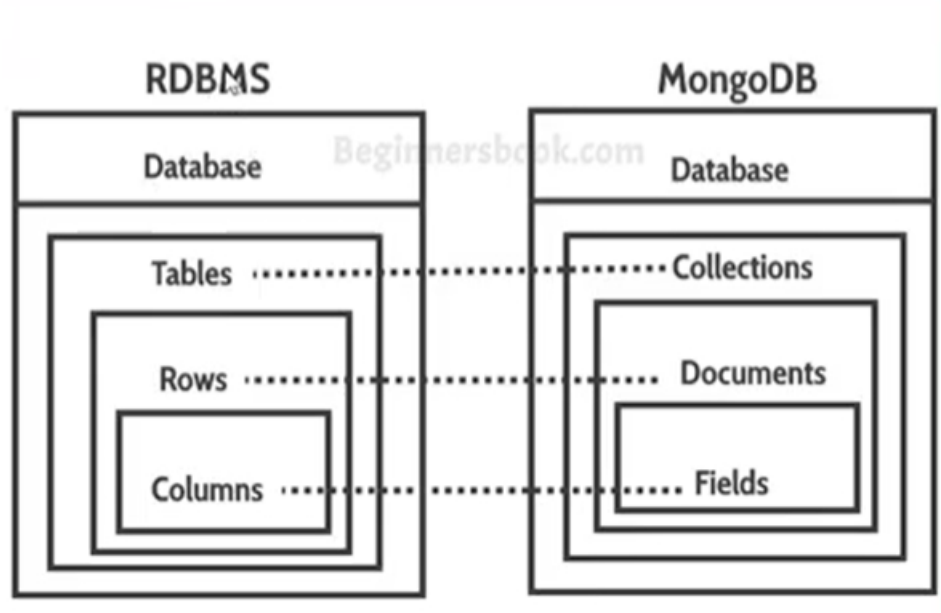
4. How the terms in MongoDB are different from SQL?

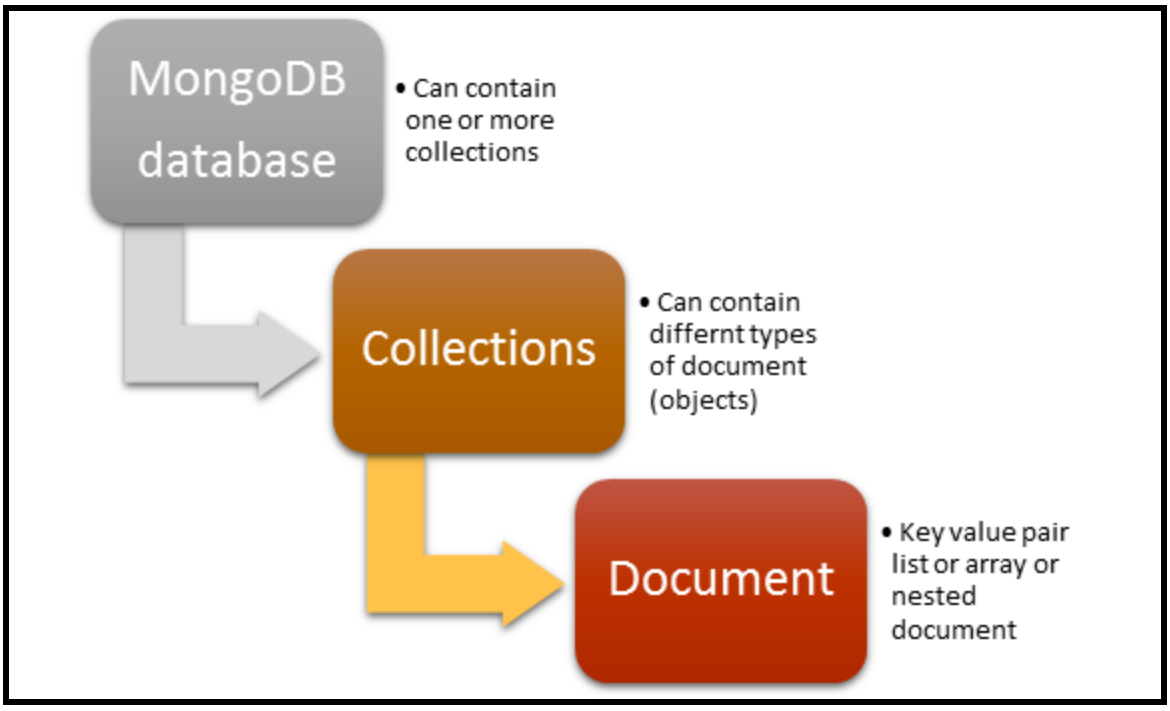
Or

Difference between SQl and NoSQL

Ans:







5.What is JSON?

JSON, or **JavaScript Object Notation**, is a simple, **readable data structure format**. As an alternative to XML, it is primarily used to transmit data between a server and a web application

JSON is made up of two main components: keys and values. They form a key/value pair when combined.
Key: A key is a string which is enclosed in quotation marks.
Value: A value can be a string, a number, a boolean expression, an array, or an object.

Key/Value Pair: A key value pair has a specific syntax, with the key coming first, followed by a colon, and then the value. Key/value pairs are separated by commas.

If you wanted to start modelling one of the rows of the table in the example, such as:

name	quantity	size	status	tags	rating
notebook	50	8.5×11,in	A	college-ruled,perforated	8

This can be done with the name and quantity fields. These fields would look like this in JSON:

```
{"name": "notebook", "qty": 50}
```

6.Define Collections and documents

A **collection** is **simply a grouping of documents** that have the same or a similar purpose.

A document is a representation of a ***single entity of data*** in the ***MongoDB database***.

7.List any four MongoDB Data types

Table 11.1 **MongoDB data types and corresponding ID number**

Type	Number
Double	1
String	2
Object	3
Array	4
Binary data	5
Object id	7
Boolean	8
Date	9
Null	10
Regular Expression	11
JavaScript	13
JavaScript (with scope)	15
32-bit integer	16
Timestamp	17
64-bit integer	18
Decimal126	19
Min key	-1
Max key	127

7.What is Normalizing Data?

Data normalization is the **process of organizing documents and collections** to minimize **redundancy and dependency.**

Advantages of Normalizing the data

- The **database size** will be **smaller** because **only a single copy of an object will *exist*** in its own collection instead of being duplicated on **multiple objects in a single *collection***.
- Also, if you ***modify the information in the subobject frequently***, you only need **to modify a single instance rather than every record in the object's collection** that has that subobject.

Disadvantages of Normalizing the data

- A major disadvantage of normalizing data is that ***when looking up user objects that require the normalized subobject***, a ***separate lookup must occur to link the subobject***.
- This can result in ***a significant performance hit*** if you are ***accessing the user data frequently***.

8.Define Capped Collections?

Definition

A **capped collection** is a collection that has **a fixed *size***.

When a new document that ***exceeds the size of the collection*** needs to be written to a collection, the oldest document in the collection is ***deleted and the new document is inserted***.

Capped collections work great for objects that have a ***high rate of insertion, retrieval, and deletion***.

Creating Capped Collection

To **create a capped collection**, we use the **normal createCollection** command but with capped option as true and specifying the maximum size of collection in bytes.

```
>db.createCollection("cappedLogCollection",{capped:true,size:10000})
```

In addition to collection size, we can also **limit the number of documents in the collection** using the max parameter −

In addition to collection size, we can also limit the number of documents in the collection using the **max** parameter −

```
>db.createCollection("cappedLogCollection",{capped:true,size:10000,max:1000})
```

9. What is Atomic Operations

Complex series of database operations in which either all or none of the operations occur.

10.What is TTL? How TTL can be implemented in MongoDB?

TTL or time-to live value for documents in each of your collections.

- (i)One way is to implement code in your application **to monitor and clean up old data.**
- (ii)Another way is to use the MongoDB TTL setting on a collection, which allows you to define **a profile where documents are automatically deleted after a certain number** of seconds or at **a specific clock time**.
- For collections where you only need **the most recent documents**, you can implement a **capped collection** that automatically keeps the size of the collection small.

11.List the things to be consider while designing a MongoDB Database

Two more important things to consider when designing a MongoDB database**(i) Data usability(ii)Performance.**

12.Write a commond to start and stop MongoDB?

Starting MongoDB

Starting MongoDB with a `port` and `dbpath` parameters:

```
mongod -port 28008 -dbpath <mongo_data_location>/data/db
```

Stopping MongoDB

```
use admin
db.shutdownServer()
```

13.What is Mongo Shell and list the commads used in Mongo Shell?

The **MongoDB shell** is an **interactive shell** provided with **MongoDB** that allows you to **access, configure, and administer MongoDB databases, users, and much more**.

- **help <option>:** Displays syntax help for MongoDB shell commands. The `option` argument allows you to specify an area where you want help.
- **use <database>:** Changes the current `database` handle. Database operations are processed on the current database handle.
- **db.help:** Displays help options for the database methods.
- **show <option>:** Shows a list based on the `option` argument. The value of `option` can be:
  - **dbs:** Displays a list of databases.

14.Write Mongodb command to list the users in Admin Database

use admin

show users

```
>
> use admin
switched to db admin
> show users
{
        "_id" : ObjectId("529e71927c798d1dd56a63d9"),
        "user" : "dbadmin",
        "pwd" : "78384f4d73368bd2d3a3e1da926dd269",
        "roles" : [
                "readWriteAnyDatabase",
                "dbAdminAnyDatabase",
                "clusterAdmin"
        ]
}
{
        "_id" : ObjectId("52a098861db41f82f6e3d489"),
        "user" : "useradmin",
        "pwd" : "0b4568ab22a52a6b494fd54e64fcee9f",
        "roles" : [
                "userAdminAnyDatabase"
        ]
}
>
```

Figure 12.3 Listing users on the admin database

15.What is MongoDB Compass?

MongoDB Compass is a ***GUI based tools (unline MongoDB Shell) to interact with local or remote MongoDB server and databases***.

Use Compass to visually explore your data, run ad hoc queries, perform CRUD operations, and view and optimize your query performance.

16.How will you count the number of users in admin Database?

```
use admin
cur = db.system.users.find()
cur.count()
```

17.Write the command to create Database Administrator and User Administrator?

Creation of Database Administrator

```
use admin
db.createUser( { user: "dbadmin",
    pwd: "test",
    roles: [ "readWriteAnyDatabase", "dbAdminAnyDatabase", "clusterAdmi
```

Creation of User Administrator

```
use admin
db.createUser( { user: "<username>",
    pwd: "<password>",
    roles: [ "userAdminAnyDatabase" ] } )
```

difference between two most popular SQL and NoSQL Dbs-

MySQL	MongoDB
Matured or stable	Its new and updated frequently
It follows tabular structure	It follows document structure like JSON format
It needs a proper schema	Its flexible in nature
Managing complex relations among different tables is easy	Its not that great in complex managing relationship
Its scales vertically	Horizontaly scalable

18.Write the command to create user in MongoDB.

Ans: The user accounts are created using the **createUser()** method.

```
use test
db.createUser( { user: "testUser",
    pwd: "test",
    roles: [ "readWrite", "dbAdmin" ] } )
```

19.How remove users from MongoDB ?

Ans: For removing the user, the db.dropUser() is used.

db.dropUser(username)

20 .How to create collection in MomgoDB ?

We can create collection explicitly using createCollection command

Syntax

db.createCollection(name.options)

Where,

name is the name of collection options is an optional field. This field is used to specify some parameters such as maximum number of documents and so on.

21.How to insert document in MongoDB?

There are 2 methods to insert documents into a MongoDB database.

(i)insertOne() (ii) insertMany()

```
db.collection.insertOne(
   <document>,
   {
     writeConcern: <document>
   }
)
```

Syntax

The `insertMany()` method has the following syntax:

```
db.collection.insertMany(
   [ <document 1> , <document 2>, ... ],
   {
      writeConcern: <document>,
      ordered: <boolean>
   }
)
```

Example

```
db.posts.insertOne({
  title: "Post Title 1",
  body: "Body of post.",
  category: "News",
  likes: 1,
  tags: ["news", "events"],
  date: Date()
})
```

```
db.posts.insertMany([
  {
    title: "Post Title 2",
    body: "Body of post.",
    category: "Event",
    likes: 2,
    tags: ["news", "events"],
    date: Date()
```

```
},
{
  title: "Post Title 3",
  body: "Body of post.",
  category: "Technology",
  likes: 3,
  tags: ["news", "events"],
  date: Date()
},
{
  title: "Post Title 4",
  body: "Body of post.",
  category: "Event",
  likes: 4,
  tags: ["news", "events"],
  date: Date()
} ])
```

22.List the methods to create connections in MongoDB?

(i) To create **an instance of the MongoClient** object.

(ii) uses a connection string to connect.

23.Define WriteConcern in MongoDB?

- Write concern **describes the guarantee that the MongoDB connection provides** when **reporting on the success of a write operation**.
  The strength of the **write concern determines the level of guarantee**

Table 13.1 Write concern levels for MongoDB connections	
Level	**Description**
`-1`	Ignores network errors.
`0`	No write acknowledgement is required.
`1`	Write acknowledgement is requested.
`2`	Write acknowledgement is requested across primary and one secondary server in the replica set.
`majority`	Write acknowledgement is requested across a majority of servers in the replica set.

24. Explain the MongoDBClient in detail.

Ans: The MongoDBClient object provides interactions to connect to the database. Following are some commonly used method of MongoDBClient object

Method	Purpose
connect(url,options)	This method is used to connect to the MongoDB using the specified url
close (force, callback)	This method closes the db and underlying connections
db(dbname, options)	It creates a new DB instance sharing the strent socket connections
isconnected(options)	It checks if MongoClient is connected

25. What is Map reduce?

Map reduce is a **data processing programming model** that helps in performing operations on large data sets and produce aggregate results Map reduce is used for large volume of data.

The syntax for map reduce is

```
db.collection.mapReduce(
  function() {emit(key,value);},  //map function
  function(key,values) {return reduceFunction}, {   //reduce function
    out: collection,
```

```
    query: document,
    sort: document,
    limit: number
  }
)
```

Where

1) **map function**: It uses emit() function in which it takes two parameters key and value key. Here the key is on which we make groups(such as group by name, or age) and the second parameter is on which aggregation is performed like avg(), sum() is calculated on.

2) **reduce function**: This is a function in which we perform aggregate functions like avg()sum()

3) out: It will specify the collection name where the result will be stored.

4) query: We will pass the query to filter the resultset.

5) sort: It specifies the optional sort criteria.

6) limit: It specifies the optional maximum number of documents to be returned.

Part-B

1.What is NoSQL? What is the need for it? Enlist the various features of NoSQL?

Definition

- ❖ Why NoSQL
- ❖ Features of NoSQL

2. List and explain various features of MongoDB.

- ❖ Definition
- ❖ Advantages
- ❖ Why Mongodb
- ❖ Understanding collections
- ❖ Understanding documents
- ❖ MongoDB datatypes
- ❖ Planning Data Model

- Data Normalization
- DeNormalizing
- Capped Collections
- Understanding Atomic write operations
- Considering document Growth
- Identifying Indexing, Sharding, and Replication Opportunities
- Deciding on Data Life Cycles
- Considering Data Usability and Performance

3.How to Build MongoDb Environment in detail

Building MongoDB Environment

- Installing MongoDB server
- Enterprise version
- Installation and setup process
- Starting MongoDB
  - Mongod
- Stopping MongoDB
- Accessing MongoDB from the Shell Client
- Understanding MongoDB Shell command
- Understanding MongoDB Shell Methods
- Understanding Command Parameters and Results
- Scripting the MongoDB Shell

4. Explain how to create users in MongoDB? Also enlist and explain the various roles of the users.

- Creating User Account
- Listing the user names
- Counting the no of user
- Assigning roles to each user
- Removing Users

5.What is Access Control?.Explain with a suitable example how to create database admininstrator account in MongoDB?

- Definition

- Role of User Administrator
- Role of Database Administrator
- Creating a User Administrator Account
- Creating a Database Administrator Account

6. What are the steps to Maintain the databases in MongoDB.

- Displaying a List of Databases
- Changing the Current Database
- Creating Databases
- Deleting Databases
- Copying Databases

7.Explain how documents are created and updated in MongoDB?

- Define documents
- Creation of documents
- Updation of doucuments
- Deletion of documents

8)Explain insertion and deletion database operation using Node JS ?

Adding the MongoDB Driver to Node.js

Connecting to MongoDB from Node.js

Understanding the Write Concern

Connecting to MongoDB from Node.js Using the MongoClient Object

9.What is the purpose of MapReduce explain with a suitable example?

Definition

- Purpose of MapReduce
- Components Of MapReduce
- Example
- Program

UNIT-4

EXPRESS AND ANGULAR

PART-A

1) Define Express.

- ❖ Express, often referred to as "Express.js" or simply "Express," is a popular and minimalist web application framework for Node.js.
- ❖ It provides a robust set of features and tools for building web and mobile applications, making it easier to develop server-side applications and APIs (Application Programming Interfaces).

2) What are the features of Express.

1. **Middleware:** Express uses middleware to handle various tasks in the request-response cycle.
   Middleware functions can perform tasks like parsing request bodies, handling authentication, logging, and more. Express allows developers to use built-in middleware or create custom middleware functions to handle specific requirements.

2. **Routing:** Express provides a simple and flexible routing system that allows developers to define routes for different HTTP methods (GET, POST, PUT, DELETE, etc.).

   These routes determine how the application responds to client requests and enable the organization of application logic.

3) What is Routing?

- ❖ Routing in Express is the process of defining how your **application responds to different HTTP requests** for specific URLs (Uniform Resource Locators).
- ❖ Express provides a **flexible and easy-to-use routing system** that allows you to map URLs to specific actions or controllers in your application.

4) What is the command to install Express JS?

npm install express

5) Write an application program using Express JS for displaying "Welcome message" on the webpage.

```
const express = require('express');
```

```
const app = express();
const port = 3000; // You can change the port number if needed
// Define a route to display the "Welcome message"
app.get('/', (req, res) => {
  res.send('<h1>Welcome to Express.js</h1><p>This is a simple Express.js web application.</p>');
});
// Start the server
app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

6) What is the syntax of Express module?

Syntax

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

Example code:

```
const express = require('express');
const app = express();
const port = 3000;
app.get('/', (req, res) => {
  res.send('Hello, Express!');
});
app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

7) What are the methods for implementing an express module? Give an example.

In Express.js, you can implement **various methods or HTTP request methods** to define routes and handle different types of HTTP requests.

Express provides methods that correspond to common HTTP request methods such as **GET, POST, PUT, DELETE**, and more. These methods are used to create routes that respond to specific HTTP requests.

8.Give the syntax for performing Routing using express.

```
// Route handler code for GET request to the root URL
app.get('/', (req, res) => {  });
// Route handler code for POST request to create URL
 app.post('/create', (req, res) => { });
  // Route handler code for PUT request to /update/:id URL
app.put('/update/:id', (req, res) => {
});
  // Route handler code for DELETE request to /delete/:id URL
app.delete('/delete/:id', (req, res) => {
});
```

9.What is Request object? Give the syntax for creating a RequestObject.

In Express.js, the `req` (request) object represents the **HTTP request made by a client to your server**.

It contains information about the request, such as headers, parameters, URL, and request body.

You can access this object within your **route handlers to process incoming requests** and extract data from them.

```
const express = require('express');
```

```
const app = express();
  const userAgent = req.headers['user-agent'];
  const queryParam = req.query.paramName;
  const requestMethod = req.method;
  const requestUrl = req.url;
```

9.List any five methods of Response Object?

- res.send()
- res.status()
- res.json()
- res.redirect()
- res.render()

10.What is Response Object?Give the syntax for creating a ResponseObject.

- The `res` (response) object in Express.js represents the **HTTP response that your server sends back to the client** after handling a request.
- It provides various methods and properties to set headers, send data, and control the response behavior. Here are five common methods of the `res` object in Express

```
const express = require('express');
const app = express();
// Define a route that handles a GET request
app.get('/', (req, res) => {
  // Use the res object to send a response to the client
  res.status(200).send('Hello, Express!');
});
// Start the Express server
app.listen(3000, () => {
  console.log('Server is running on port 3000');
```

});

11.Define Angular framework.

- Angular is a popular **open-source front-end web application framework** developed and maintained by Google and a community of individual developers and corporations.
- It is designed to simplify the development of **dynamic, single-page web applications (**SPAs) and provide a structured and organized way to build complex, client-side applications.

12.What is TypeScript? State two advantages.

TypeScript is an **open-source programming language** developed and maintained by Microsoft. It is a s**trict syntactical superset of JavaScript,** which means that any valid JavaScript code is also valid TypeScript code

1. Static Typing
2. Advanced Features and Tooling

13.What are the features of TypeScript?

1) **Module System:** TypeScript uses a module system that helps organize code into reusable and maintainable units. It supports both CommonJS and ES6 modules.
2) **IDE Support:** Most popular integrated development environments (IDEs) and code editors provide excellent support for TypeScript. Features like autocompletion, type checking, and inline documentation enhance developer productivity.

14. How to write a class in TypeScript?

```
class Person {
  // Properties
  firstName: string;
  lastName: string;
  // Constructor
  constructor(firstName: string, lastName: string) {
```

```
    this.firstName = firstName;
    this.lastName = lastName;
  }
  // Method
  getFullName(): string {
    return `${this.firstName} ${this.lastName}`;
  }
}
// Creating an instance of the class
const person1 = new Person('John', 'Doe');
// Accessing properties and calling methods
console.log(person1.firstName); // Output: John
console.log(person1.getFullName()); // Output: John Doe
```

15.What is the use of ngModule in Angular?

- Angular modules consolidate components, directives and pipes into cohesive blocks of functionality.
- For defining the module we use the ngModule.
- Every Angular application has at least one NgModule class, the root module, which is conventionally named AppModule and resides in a file named app.module.ts.

16.What are the two types of data bindings in Angular?

There are two types of data bindings-

1. **Property binding:** This type of binding allows to pass the interpolated values from application data to HTML. The interpolated values are specified in {{ and }} bracket pair. For instance - the student.name is interpolating value.

```
<p>Name: {{ student.name}}</p>
```

2. **Event binding:** Event binding is used to capture events on the user's end on the app and respond to it in the target environment by updating the application data.

17.What is Directives?List the types of Directives?

Directives are JavaScript classes with metadata that defines the structure and behavior. Directives provide the majority of UI functionality for Angular applications.

There are three major types of directives:

Components: A component directive is a directive that incorporates an HTML template with JavaScript functionality to create a self-contained UI element that can be added to an Angular application as a custom HTML element.

Components are likely to be the directives you use the most in Angular.

Structural:. Structural directives allow you to create and destroy elements and components from a view.

Attribute: An attribute directive changes the appearance and behavior of HTML elements by using HTML attributes.

18.What is Dependency Injection?

Dependency injection is a process in which a component defines dependencies on other components. When the code is initialized, the dependent component is made available for access within the component. Angular applications make heavy use of dependency injection.

19.List the rules to implement Angular?

- The view acts as the official presentation structure for the application. Indicate any presentation logic as directives in the HTML template of the view.
- If you need to perform any DOM manipulation, do it in a built-in or custom directive JavaScript code—and nowhere else.
- Implement any reusable tasks as services and add them to your modules by using dependency injection.
- Ensure that the metadata reflects the current state of the model and is the single source for data consumed by the view.

- ❖ Define controllers within the module namespace and not globally to ensure that your application can be packaged easily and avoid overwhelming the global namespace.

20.List some Command Line Interface commands in Angular?

Table 21.1 Angular CLI Command Options

Command	Alias	Purpose
`ng new`		Creates a new Angular application
`ng serve`		Builds and runs the angular application for testing
`ng eject`		Makes the webpack config files available to be edited
`ng generate component [name]`	`ng g c [name]`	Creates a new component
`ng generate directive [name]`	`ng g d [name]`	Creates a new directive

PART B QUESTIONS

1.Write the steps to install Express in Node.js?

Getting Started with Express

npm install express

Configuring Express Settings

Table 18.1 Express application settings

Setting	Description
`env`	Defines the environment mode string, such as `development`, `testing`, and `production`. The default is the value of `process.env.NODE_ENV`.
`trust proxy`	Enables/disables reverse proxy support. The default is disabled.
`jsonp callback name`	Defines the default callback name of JSONP requests. The default is `?callback=`.
`json replacer`	Defines the JSON replacer callback function. The default is `null`.
`json spaces`	Specifies the number of spaces to use when formatting JSON response. The default is `2` in development, `0` in production.

Starting the Express Server

2.How to handle the HTTP request by the HTTP Server?

Define Route

A route is simply a definition that describes how to handle the path portion of the URI in the HTTP request to the Express server.

- ❖ Configuring Routes
- ❖ Implementing Routes
- ❖ Applying Parameters in Routes
- ❖ Applying Route Parameters Using Query Strings
- ❖ Applying Route Parameters Using Regex
- ❖ Applying Route Parameters Using Defined Parameters
- ❖ Applying Callback Functions for Defined Parameters

3.Write Short note on Angular?

- ❖ Definition
- ❖ Why Angular?
- ❖ Modules
- ❖ Directive
- ❖ Data Binding
- ❖ Dependencies
- ❖ Services
- ❖ Angular CLI

4.Explain the building blocks to create Angular applications.?

Angular Components

```
Import {Component} from '@angular/core';
@Component({
     selector: 'my-app',
     template: '<p>My Component</p>'
})
Export class AppComponent{
    Title = 'Chapter 1 Example';
}
```

- ❖ Defining a selector
- ❖ Building a Template
- ❖ Using Inline CSS and HTML in Angular Application

```
01 import { Component } from '@angular/core';
02
03 @Component({
04   selector: 'app-root',
05   template: `
06     <span>Hello my name is Brendan</span>
07   `,
08   styles:[`
09     span {
10       font-weight: bold;
11       border: 1px ridge blue;
12       padding: 5px;
13     }
14   `]
15 })
16 export class AppComponent {
17   title = 'Chapter 22 Intro';
18 }
```

5.Write expression for Basic Strings and Numbers with Simple Math Operations in an Angular Template?

```
01 import { Component } from '@angular/core';
02
03 @Component({
04   selector: 'app-root',
05   template: `
06     <h1>Expressions</h1>
07     Number:<br>
08     {{5}}<hr>
09     String:<br>
10     {{'My String'}}<hr>
11     Adding two strings together:<br>
12     {{'String1' + ' ' + 'String2'}}<hr>
13     Adding two numbers together:<br>
14     {{5+5}}<hr>
15     Adding strings and numbers together:<br>
16     {{5 + '+' + 5 + '='}}{{5+5}}<hr>
17     Comparing two numbers with each other:<br>
18     {{5===5}}<hr>
19   `,
20 })
21 export class AppComponent {}
```

6.What is Data Binding in Angular explain in detail with a program.

Data binding is a key feature in Angular, a popular JavaScript framework for building dynamic web applications.

It allows you to establish a connection between the data in your application's JavaScript code (component) and the DOM (Document Object Model) elements in your HTML templates. There are four types of data binding in Angular:

1. **Interpolation** (`{{...}}`)**: This is a one-way data binding method that allows you to display component data in the HTML template.

2. **Property Binding** (`[property]="..."`)**: This is another form of one-way data binding that sets a property value of an HTML element to a component property.

3. Event Binding (`(event)="..."`)**: Event binding lets you listen to DOM events (like clicks, mouse movements, and keyboard inputs) and trigger methods in your component when those events occur.

4.Two-Way Data Binding (`[(ngModel)]`)**: This type of binding combines property binding and event binding to establish a two-way connection between a form control element (input, select, textarea) and a component property. It allows changes in the template to be reflected in the component and vice versa.

Let's explore these types of data binding in more detail with a simple Angular program.

Example of Data Binding in Angular:

Consider a simple Angular application that displays a user's name and allows them to update it. We'll demonstrate all four types of data binding in this example.

1. Create an Angular Component:

ng generate component user

2.Modify the `user.component.ts`:

```typescript
// user.component.ts
import { Component } from '@angular/core';
@Component({
  selector: 'app-user',
  templateUrl: './user.component.html',
})
```

```
export class UserComponent {
  userName = 'John Doe';

  updateUserName(newName: string) {
    this.userName = newName;
  }
}
```

3.Modify the `user.component.html`:

```html
<!-- user.component.html -->
<h2>User Profile</h2>
<!-- Interpolation -->
<p>Welcome, {{ userName }}!</p>
<!-- Property Binding -->
<input [value]="userName" [placeholder]="'Enter new name'" #nameInput />
<!-- Event Binding -->
<button (click)="updateUserName(nameInput.value)">Update Name</button>
<!-- Two-Way Data Binding (ngModel) -->
<input [(ngModel)]="userName" />
```

4. Update `app.module.ts` to enable Two-Way Data Binding:**

```typescript
// app.module.ts
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms'; // Import FormsModule
```

```
import { AppComponent } from './app.component';
import { UserComponent } from './user/user.component';

@NgModule({
  declarations: [AppComponent, UserComponent],
  imports: [BrowserModule, FormsModule], // Add FormsModule to imports
  bootstrap: [AppComponent],
})
export class AppModule {}
```

5. Use the `UserComponent` in your `app.component.html`:**

```
```html
<!-- app.component.html -->

<h1>Data Binding in Angular</h1>
<app-user></app-user>
``
```

6. Run your Angular application:**

```
ng serve
```

This will start the development server, and you can view the application in your browser.

7.Explain in Built-in directives in detail

- Understanding Directives
  - Directives are a combination of Angular template markup and supporting TypeScript code.
  - Angular directive markups can be HTML attributes, element names, or CSS classes.
- Using Built-in Directives

  Component: A directive with a template
  Structural: A directive that manipulates elements in the DOM
```

Attribute: A directive that manipulates the appearance and behavior of a DOM element

- ❖ Components Directives
- ❖ Structural Directives
- ❖ Attribute Directives

UNIT-5

REACT

QUESTION BANK

PART-A

1) What is MEAN & MERN stack?

The **MEAN (MongoDB, Express, AngularJS, Node.js**) stack was one of the early open-source stacks that epitomized this shift toward SPAs and adoption of NoSQL.

AngularJS, a front-end framework based on the Model View Controller (MVC) design pattern, anchored this stack. MongoDB, a very popular NoSQL database, was used for persistent data storage.

Node.js, a server-side JavaScript runtime environment, and ***Express, a web-server built on Node.js, formed the middle-tier***, or the web server.

Not exactly competing, but React, an alternate front-end technology created by Facebook, has been gaining popularity and offers an alternative to AngularJS. It thus replaces the "A" with an "R" in MEAN, to give us the MERN Stack.

React is not a full-fledged MVC framework. It is a JavaScript library for building user interfaces, so in some sense, it's the View part of the MVC.

2.What is React Router?

React gives us only the View rendering capability and helps manage interactions in a single component.

When it comes to transitioning between different views of the component and keeping the browser URL in sync with the current state of the view, we need something more.This **capability of managing URLs and history** is called **routing.**

3. What is Bootstrap and React-BootStrap?

Bootstrap is a free, open source front-end development framework for the creation of websites and web apps. Designed to enable responsive development of mobile-first websites, Bootstrap provides a collection of syntax for template designs.

The most popular CSS framework, has been adapted **to React and the project is called ReactBootstrap**.

This library not only gives us most of the Bootstrap functionality, but the components and

widgets provided by this library also give us a wealth of information on how to design our own widgets and components

4. **What is React and which layer of web application is it responsible for?**

- React anchors the MERN stack. In some sense, **it is the defining component of the MERN stack.**
- React is an open-source JavaScript library maintained by **Facebook that can be used for creating views rendered** in HTML. Unlike AngularJS, React is not a framework.
- It is a library. Thus, it does not, by itself, dictate a framework pattern such as the MVC pattern. You use React to render a view (the V in MVC),

5. **State the role of MongoDB in MERN stack.**

MongoDB plays a crucial role in the **MERN (MongoDB, Express.js, React, Node.js) stack** as the "M" component, which stands for the database layer.

MongoDB is a NoSQL database system that is used to store and manage the application's data. Here are the key roles of MongoDB in the MERN stack:

React, the front-end library in the MERN stack, can make **HTTP requests to the Express.js backend,** which in turn communicates with MongoDB to fetch and update data. This allows for a seamless flow of data between the client and the database.

1. Data Storage:
2. Scalability:
3. Querying and Indexing:
4. Flexibility:
5. JSON-Like Documents:
6. React Integration:

6. **Define Virtual DOM.**

- The Virtual DOM is a l**ightweight copy or representation of the real DOM.** It is a JavaScript object that mirrors the structure of the actual DOM.
- When changes need to be made to the UI, developers don't directly manipulate the real DOM. Instead, they update the Virtual DOM, which is much faster because it's a purely in-memory operation and doesn't involve the browser rendering engine.

- The Virtual DOM keeps **track of the differences (or "diffs") between its current state and the desired state.**

7.What is JFX?

- "JFX" typically refers to **"JavaFX," which is a Java-based framework and library** for building desktop and Rich Internet Applications (RIAs).
- JavaFX provides a **platform-independent way to create interactive** and visually appealing user interfaces (UIs) for various types of applications. Here are some key points about JavaFX:

8.Define components in react?

In React, components are the **building blocks of a user interface.**
They are reusable, self-contained pieces of code that encapsulate a part of the user interface and its behaviour.
React applications are typically composed of **multiple components that work together to create complex user interfaces.**
There are two main types of components in React: functional components and class components.

9.What is One way Data Binding?

One-way data binding is a data flow pattern commonly used in **web development frameworks and libraries,** including React and Angular.

In one-way data binding, data flows in one direction, from the data source (typically a component or a model) to the user interface (UI) or view.

Changes to the data source are reflected in the UI, but changes in the UI do not directly affect the data source. This unidirectional flow of data simplifies application architecture and helps prevent unintended side effects.

10.What is Functional Component?

- In React, a functional component is a **JavaScript function that represents a reusable UI component.**
- Functional components are also known as **stateless components or presentational components.**
- They are primarily responsible for rendering the UI based on the data provided to them via props (properties). Functional components do not have their own internal state or lifecycle methods like class components; instead, they receive data and return JSX (JavaScript XML) to describe how the UI should look.

11.What is Class Component?

- In React, a class component is a JavaScript class **that represents a reusable UI component.**
- Class components are also known as stateful components because they have their own internal state and can use lifecycle methods for managing the component's behavior over time.
- Class components were the primary way of creating components in React before the introduction of hooks.

12.Difference between Functional and Class Components.

- Functional components are also known as stateless components or presentational components.
- Functional components do not have their own internal state or lifecycle methods like class components; instead, they receive data and return JSX (JavaScript XML) to describe how the UI should look.
- Class components are also known as stateful components because they have their own internal state and can use lifecycle methods for managing the component's behavior over time.
- Class components were the primary way of creating components in React before the introduction of hooks

.13.What is Inter Components Communication?

In React, there are several ways to achieve inter-component communication, allowing different components in your application to exchange data or trigger actions.

1. Props (Parent to Child)
2. Callback Functions (Child to Parent)
3. Context API (Global State)
4. Redux (External State Management)

14.What is React state?

In React, "state" refers to an internal data structure that allows components to store and manage information that can change over time.

State is a fundamental concept in React and is often used to keep track of dynamic data that needs to be displayed in a component. When the state of a component changes,

Here are the key aspects of React state:

1. Class Components:

- `this.state` object. You initialize the state in the component's constructor, and then you can update it using the `this.setState()` method.

2. Functional Components with Hooks:

- In functional components, state can be managed using the `useState` hook. The `useState` hook returns an array with two elements: the current state value and a function to update it.

3. Immutability:

- In React, you should never directly modify the state object. Instead, you use the provided update function (`setState` in class components or the function returned by `useState`) to create a new state object that reflects the changes you want.

- React relies on immutability to determine when to re-render components.

4. Asynchronous Updates:

- State updates in React are asynchronous, which means that React batches multiple updates and applies them in a single pass for performance reasons.

- To ensure that the next state depends on the previous state, you can use the functional form of `setState` (class components) or the functional update form of `useState` (functional components).

15)What are the Differences between Props and State.

Props and state are two fundamental concepts in React used for managing data and passing information between components. While they both serve similar purposes, they have some key differences:

Props (Properties):

1. Source of Data: Props are used for passing data from a parent component (the component that renders another component) to a child component (the component being rendered).

2. Immutability: Props are immutable, which means that they cannot be modified or changed by the child component that receives them. They are read-only.

State

1. Source of Data: State is used for managing and storing data that can change within a component. It is internal to a component and can be modified by the component itself.

2. Mutability: State is mutable, meaning a component can change its own state using `this.setState()` (class components) or the state updater function returned by the `useState` hook (functional components).

16.Draw the sequence diagram for browser rendering and serverrendering

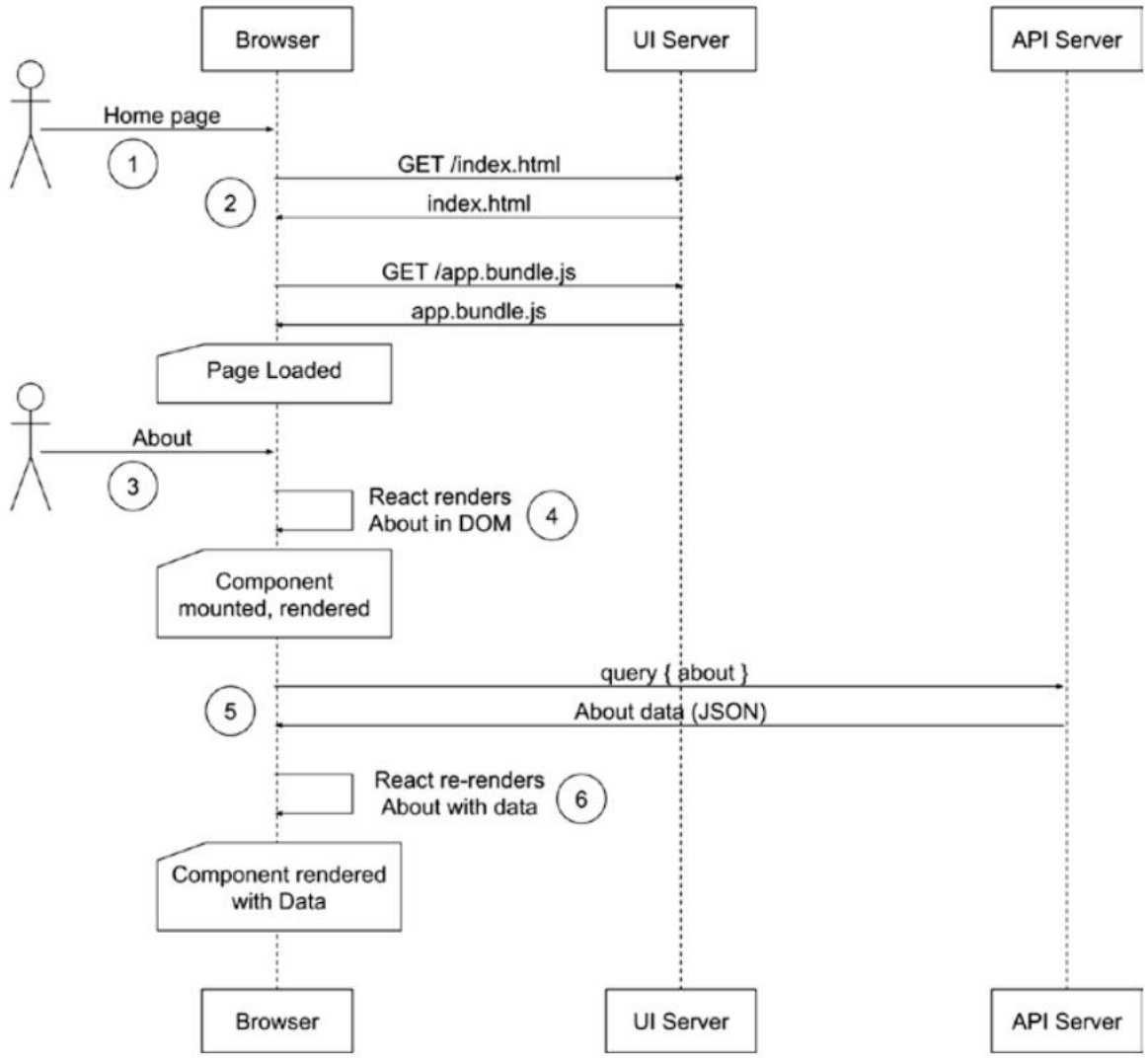


Figure 12-2. *Sequence diagram for browser rendering*

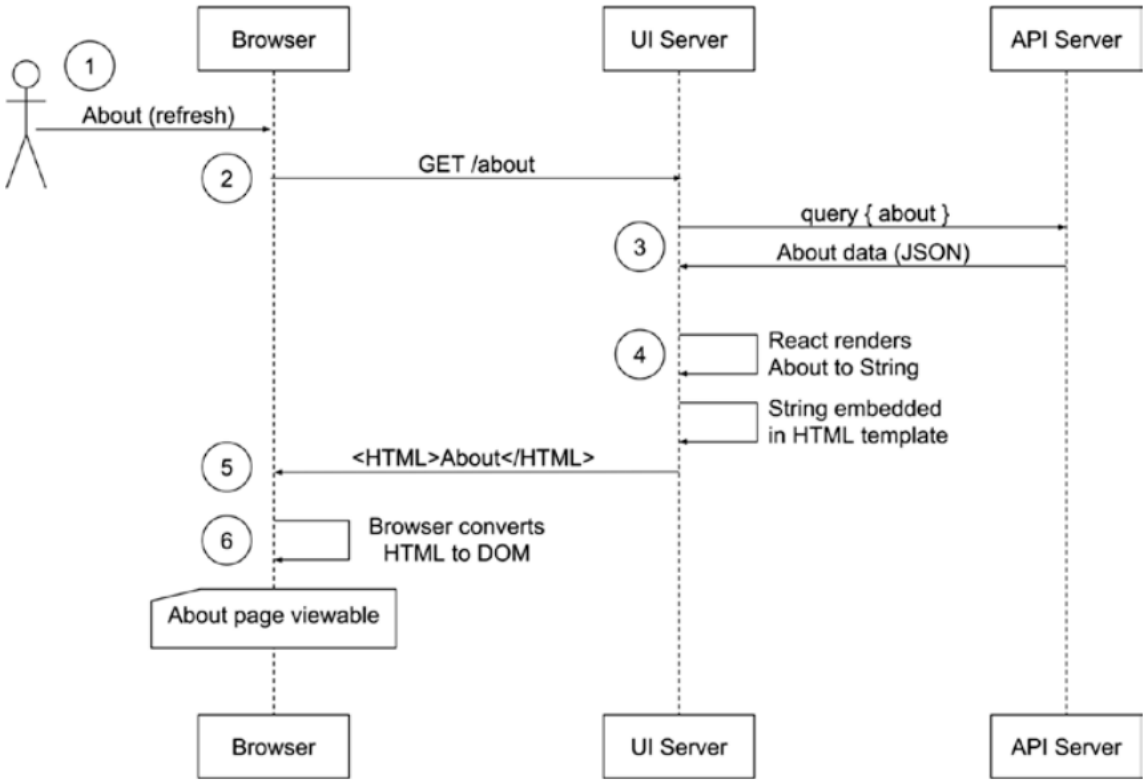


Figure 12-4. *Sequence diagram for server rendering*

17.What are webpack?

Webpack in react is a JavaScript module bundler that is commonly used with React to bundle and manage dependencies.

It takes all of the individual JavaScript files and other assets in a project, such as images and CSS, and combines them into a single bundle that can be loaded by the browser.

18. What is modularization in react?

Modularity helps in code organization by splitting the code into smaller parts. A modular architecture involves breaking down a program into sub-programs called modules. Note: The module refers to a component or set of components that perform a specific task.

19. What is RESTAPI?

REST API stands for **Representational State Transfer application programming** interface, sometimes referred to as the RESTful API, and it's the **primary interface** that is used by React.js developers which allows ***API connections between different parts of an application or service*** over the internet.

20.What is Hydration?

Hydration is the process of using client-side JavaScript to add application state and interactivity to server-rendered HTML.

It's a feature of React, one of the underlying tools that make the Gatsby framework.

Gatsby uses hydration to transform the static HTML created at build time into a React application.

PART-B

1.Explain in detail the MERN stack?

MERN Components

MERN stack and a few other libraries and tools that we will be using to build your web application.

Features of MERN

React

React is an open-source JavaScript library maintained by Facebook that c an be used for creating views rendered in HTML.

Why Facebook Invented React

- Imperative.
- Declarative

Component-Based

No Templates

Isomorphic

Node.js Modules

Node.js and npm

Node.js Is Event Driven

Express

MongoDB

NoSQL

Document-Oriented

JavaScript Based

React-Router

React-Bootstrap

Webpack

2.Explain the steps of installation of ReactJS. Write a program using ReactJS.

To install ReactJS and create a basic React application, you'll need to follow these steps:

Step 1: Install Node.js and npm

ReactJS relies on Node.js and npm (Node Package Manager) for development and package management. If you don't have Node.js and npm installed, you can download and install them from the official website: https://nodejs.org/

Step 2: Create a React Application

Once you have Node.js and npm installed, you can create a new React application using `create-react-app`, which is a tool that sets up a basic React project structure with all the necessary dependencies and configurations.

Open your terminal or command prompt and run the following command:

npx create-react-app my-react-app

Replace `my-react-app` with the name you want to give to your React application. This command will create a new directory with the specified name and set up a basic React project inside it.

Step 3: Navigate to Your Project Directory

After the command has completed, navigate to your project directory:

cd my-react-app

Step 4: Start the Development Server

To start the development server and run your React application, use the following command:

npm start

This will start the development server, and your React app will be available at `http://localhost:3000` in your web browser.

Step 5: Create a React Component

Now, let's create a simple React component. React components are typically defined in `.js` or `.jsx` files.

In your project directory, open the `src` folder and create a new file called `HelloWorld.js`. Inside this file, you can define a basic React component:

```javascript
import React from 'react';

function HelloWorld() {
  return (
    <div>
      <h1>Hello, React!</h1>
      <p>This is a basic React component.</p>
    </div>
  );
}

export default HelloWorld;
```

Step 6: Use the React Component

Now, let's use the `HelloWorld` component in the `src/App.js` file, which is the main entry point for your React application:

```javascript
import React from 'react';
import './App.css';
import HelloWorld from './HelloWorld'; // Import the HelloWorld component
```

```
function App() {
  return (
    <div className="App">
      <header className="App-header">
        <HelloWorld /> {/* Use the HelloWorld component */}
      </header>
    </div>
  );
}
export default App;
```

Step 7: Start the Development Server

If the development server is not already running, start it again with `npm start`.

Step 8: View Your React App

Open your web browser and go to `http://localhost:3000`. You should see your React app displaying the "Hello, React!" message.

3.Explain the React components with a program.

. In React, components are the building blocks of your user interface. They are reusable, self-contained pieces of code that can be composed together to create complex user interfaces.

There are two main types of components in React: **functional components and class components.**

Step 1: Create a React App

Before you start, make sure you have a React app set up as explained in the previous response.

Step 2: Create a Functional Component

In your project directory, open the `src` folder and create a new file called `Counter.js`. Inside this file, create a functional component:

```javascript
import React, { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  const increment = () => {
    setCount(count + 1);
  };
  const decrement = () => {
    setCount(count - 1);
  };
  return (
    <div>
      <h2>Counter</h2>
      <p>Count: {count}</p>
      <button onClick={increment}>Increment</button>
      <button onClick={decrement}>Decrement</button>
    </div>
  );
}
export default Counter;
```

In this functional component:

- We use the `useState` hook to manage the state of the `count` variable.

- We define two functions, `increment` and `decrement`, which update the `count` state.

- The component renders the current count and two buttons to increment and decrement it.

Step 3: Use the Functional Component

Now, let's use the `Counter` component in the `src/App.js` file:

```javascript
import React from 'react';
import './App.css';
import Counter from './Counter'; // Import the Counter component
function App() {
  return (
    <div className="App">
      <header className="App-header">
        <Counter /> {/* Use the Counter component */}
      </header>
    </div>
  );
}
export default App;
```

Step 4: Start the Development Server

If the development server is not already running, start it with `npm start`.

Step 5: View Your React App

Open your web browser and go to `http://localhost:3000`. You should see your React app displaying the "Counter" component with a count and buttons to increment and decrement it.

This example demonstrates how to create a functional React component, manage state using the `useState` hook, and use that component within another component. React components can be composed and reused to build more complex user interfaces.

4.Explain Express rest API's methods in detail with an example.

1. `GET` Method

The `GET` method is used to retrieve data from a server. It should not have any side effects on the server; it's meant for read-only operations.

2. `POST` Method**

The `POST` method is used to create a new resource on the server. It typically involves sending data to the server to be processed and stored.

Example:

```javascript
const express = require('express');
const app = express();
app.use(express.json()); // Middleware for JSON request body parsing
const users = [];
app.post('/api/users', (req, res) => {
  const newUser = req.body;
  users.push(newUser);
  res.status(201).json(newUser); // 201 Created status code
});

app.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

In this example, a `POST` request to `/api/users` will create a new user resource on the server.

3. `PUT` Method**

The `PUT` method is used to update an existing resource or create it if it doesn't exist. It replaces the entire resource with the new data.

5.Explain Modularization and Webpacks in detail with the necessary commands and steps

Modularization and Webpack are essential concepts in modern JavaScript development, especially in the context of building complex web applications.
Modularization:

Modularization is the practice of breaking down a large codebase into smaller, self-contained modules. Each module focuses on a specific functionality or feature of the application. Modularization helps in organizing code, improving maintainability, and facilitating collaboration among developers.

Step 1: Create Module Files**

Step 2: Use Modules

Webpack:

Webpack is a popular JavaScript module bundler. It allows you to bundle all your modules and their dependencies into a single file (or multiple files) for use in the browser. This helps optimize the loading and execution of JavaScript in web applications.

Here are the steps to use Webpack:

Step 1: Install Webpack**

You need to install Webpack and its command-line interface (CLI) globally if you haven't already:

Step 2: Create a Webpack Configuration**

Create a `webpack.config.js` file in your project root to configure Webpack. Here's a basic configuration:

```javascript
// webpack.config.js

const path = require('path');

module.exports = {
  entry: './src/app.js',
  output: {
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist'),
```

```
  },
};
```

In this configuration:

- `entry` specifies the entry point of your application (e.g., `app.js`).

- `output` specifies the output file and path (e.g., `bundle.js` in the `dist` folder).

Step 3: Bundle Modules

Run Webpack to bundle your modules

Step 4: Include Bundled File in HTML**

In your HTML file (e.g., `index.html`), include the bundled JavaScript file:

```html
<!DOCTYPE html>
<html>
<head>
  <title>Webpack Example</title>
</head>
<body>
  <script src="dist/bundle.js"></script>
</body>
</html>
```

Step 5: Run the Application

PART-C

6.Develop an Employee Management System Application using NodeJS API.

Creating a full-fledged Employee Management System (EMS) application using a Node.js API is a complex task that involves several components, including a database for storing employee data, authentication, and frontend development. Below is a simplified example of a Node.js API for an EMS that allows you to perform basic CRUD (Create, Read, Update, Delete) operations on employee records.

Step 1: Set Up Your Project**

Create a new directory for your project and initialize a Node.js project:

mkdir employee-management-system

cd employee-management-system

npm init -y

Step 2: Install Dependencies**

You'll need the following dependencies for your project:

- `express`: To create the API and handle HTTP requests.

- `mongoose`: To interact with a MongoDB database.

- `dotenv`: To manage environment variables.

- `body-parser`: To parse JSON request bodies.

Install these dependencies using npm:

npm install express mongoose dotenv body-parser

Step 3: Set Up MongoDB**

You need to set up a MongoDB database to store employee data. You can use a local MongoDB instance or a cloud-based service like MongoDB Atlas.

Step 4: Create the Express Server**

Create an `app.js` file to set up the Express server:

```javascript

const express = require('express');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

require('dotenv').config();

const app = express();

const PORT = process.env.PORT || 3000;

**// MongoDB Connection**

mongoose.connect(process.env.MONGODB_URI, {

  useNewUrlParser: true,
```

```
  useUnifiedTopology: true,
  useCreateIndex: true,
});
// Body Parser Middleware
app.use(bodyParser.json());
// Routes
app.use('/api/employees', require('./routes/employees'));
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

Step 5: Create Employee Model

Create a `models/Employee.js` file to define the Employee model using Mongoose:

```javascript
const mongoose = require('mongoose');
const employeeSchema = new mongoose.Schema({
  firstName: String,
  lastName: String,
  email: String,
  position: String,
  department: String,
  hireDate: Date,
});
module.exports = mongoose.model('Employee', employeeSchema)
```

Step 6: Create API Routes

Create a `routes/employees.js` file to define API routes for CRUD operations on employees:

```javascript
```

```
const express = require('express');
const router = express.Router();
const Employee = require('../models/Employee');
// Create a new employee
router.post('/', async (req, res) => {
  try {
    const employee = new Employee(req.body);
    await employee.save();
    res.status(201).json(employee);
  } catch (error) {
    res.status(400).json({ error: error.message });
  }
});
// Get all employees
router.get('/', async (req, res) => {
  try {
    const employees = await Employee.find();
    res.json(employees);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
// Update an employee
router.put('/:id', async (req, res) => {
  try {
    const employee = await Employee.findByIdAndUpdate(req.params.id,
req.body, {
      new: true,
```

```
    });
    res.json(employee);
  } catch (error) {
    res.status(400).json({ error: error.message });
  }
});
// Delete an employee
router.delete('/:id', async (req, res) => {
  try {
    await Employee.findByIdAndDelete(req.params.id);
    res.json({ message: 'Employee deleted' });
  } catch (error) {
    res.status(400).json({ error: error.message });
  }
});
module.exports = router;
```

Step 7: Set Up Environment Variables**

Create a `.env` file in your project root and add the following environment variables:

```
PORT=3000
MONGODB_URI=your-mongodb-uri
```

Replace `your-mongodb-uri` with your actual MongoDB connection URI.

Step 8: Run the Application**

Start your Node.js application:

```
node app.js
```

Your Employee Management System API should now be running on the specified port (default is 3000). You can use tools like Postman or create a frontend application to interact with the API for managing employee records.

7.Explain Server-Side Rendering in detail, also Explain how it differs from Client-Side Rendering.

Server-Side Rendering (SSR) and Client-Side Rendering (CSR) are two different approaches to rendering web pages in web applications.

Server-Side Rendering (SSR):

Server-Side Rendering is a technique in web development where the server generates the HTML content for a web page and sends it to the client's browser as a fully rendered page. Here's how SSR works:

1. Client Request: When a user makes a request to a website, the server receives the request.

2. Server Processing: The server processes the request, retrieves data from databases or APIs, and generates the HTML content for the requested page.

3. HTML Sent to Client: The server sends the fully rendered HTML page as a response to the client's browser.

4. Client Presentation: The client's browser receives the HTML, displays it, and executes any associated JavaScript code.

Key characteristics of SSR:

- Improved Initial Load: SSR typically results in faster initial page loads because the client receives pre-rendered HTML from the server.

- SEO-Friendly: Search engines can easily index and rank pages because the content is available in the initial HTML response.

- Limited Client-Side JavaScript: SSR reduces the reliance on client-side JavaScript for rendering content.

Client-Side Rendering (CSR):

Client-Side Rendering is an approach where most of the rendering work is done in the client's browser using JavaScript. Here's how CSR works:

1. Client Request: The user makes a request to a website, and the server responds with a minimal HTML shell and JavaScript code.

2.JavaScript Execution: The client's browser executes the JavaScript code, which includes making API requests to fetch data.

3.Dynamic Rendering: Using the fetched data, the JavaScript code dynamically generates and updates the page content in the browser.

Key characteristics of CSR:

- Fast Subsequent Navigation: CSR can provide faster navigation between pages after the initial load because only data is fetched, and page updates are done in the client.

- Rich Interactivity: CSR allows for highly interactive web applications where the content can change without full page reloads.

- SEO Challenges: Search engines may struggle with indexing content as the initial HTML response is minimal, and the content is often loaded dynamically.

Differences Between SSR and CSR:

1.Initial Load Time:

- SSR: Faster initial load times as the server sends fully rendered HTML.

- CSR: Slower initial load times because the client needs to fetch data and render the page.

2SEO:

- SSR: SEO-friendly as search engines can easily index the pre-rendered HTML.

- CSR: SEO can be challenging as content is often loaded dynamically and may not be readily available for indexing.

3. JavaScript Dependency:

- SSR: Reduced reliance on client-side JavaScript for rendering content.

- CSR: Heavily relies on client-side JavaScript for rendering and updating content.

4. Complexity:

- SSR: Simpler client-side code but can be more complex on the server.
- CSR: More complex client-side code but simpler server-side rendering.

5. Resource Usage:

- SSR: Lower client resource usage as the server does most of the rendering.

- CSR: Higher client resource usage, especially on less powerful devices.