**Q. What is Javascript engine? Explain with examples. \* Javascript is a scripting language and is not directly understood by computer .A javascript engine is a software component that executes javascript code .these engines helps to convert out javascript program into computer understandable language . \*Javascript engines are typically developed by web vendors and every major browser has one. In a browser the javascript engine runs in concert with the rendering engine via Document Object Model.\*Following are some javascript engines:\***

**1. V8 - v8 is a javascript engine developed by the Google chrome which can run standalone because it is written in c++. This allows you to write your own program that can do everything that v8 can do and much more . your program can run ECMAScript and additional features available like file handling,database connections etc. which are available with c++ but not with javascript**

**.2. Chakra - chakra is a jscript engine developed by Microsoft .It is a proprietary software .It is used in the Internet Explorer web browser and Original edge browser but the edge was later rebuilt and now uses v8.A distinctive feature of the engine is that JIT compliles scripts on a separate CPU core ,parallel to the web browser.3.Spider Monkey -SpiderMonkey is the first javascript engine,written by Brenden Eich at Netscape Communications,later released as open-source and currently maintained by the Mozilla foundation .It is used in the Firefox web browser. 4.JavaScriptCore – JavaScriptCore(JSC)is the JavaScript engine used by Safari and MacOS.**

**What is Node.js? Write a short note on features of Node.js.**

**\*Node.js is an open-source, cross-platform JavaScript runtime environment that allows developers to execute JavaScript code server-side. \*It is built on the V8 JavaScript runtime engine, which is the same engine that powers Google Chrome. Node.js enables developers to use JavaScript for server-side scripting, allowing them to build scalable network applications quickly. \*Some of the features of Node.js are:\* 1) Asynchronous and Event driven: One of the most significant features of Node.js is its asynchronous, non-blocking I/O model. This allows Node.js to handle many simultaneous connections without the need for multi-threading.2) Single Programming Language: Node.js enables developers to use JavaScript for both client-side and server-side development. This unification of programming languages helps streamline the development process, making it more cohesive and efficient. 3) Fast Execution: Node.js is built on the V8 JavaScript engine, which compiles JavaScript directly to machine code. This results in fast execution times, making Node.js suitable for building high-performance applications. 4)Real-time Capabilities: Node.js is well-suited for building real-time applications such as chat applications, online gaming, and collaborative tools. Its event-driven architecture makes it capable of handling multiple simultaneous connections and responding in real-time. 5) Cross-Platform: Node.js is a cross-platform and can run on various operating systems, including Windows, macOS, and Linux.**

**Q. Node.js Process Model :**

**▪ Node.js runs in a single process with requests being processed on a single main thread (application thread) ▪ When a request comes in, it will be placed in an event queue. ▪ Node.js uses an event loop which continuously runs, receiving requests from :the event queue. ▪ If the request is non-blocking, (i.e. it does not involve any long-running processes or data requests), the response will be immediately prepared and then sent back to the client. ▪ If the request is blocking, (i.e. requiring I/O operations), the request will be sent to a worker thread pool. ▪ Node.js uses internal C++ thread pool to provide asynchronous I/O. ▪ The request will have an associated callback function that will fire when the request is finished and the worker thread can send the response to the event loop. This response is then sent back to the client. ▪ The single main thread (application thread) receives a request, it hands it off so that it can process other requests in the meantime. In this way, Node.js is inherently asynchronous. web server is a software application or program that facilitates communication between clients (such as web browsers) and web applications over the internet or an intranet. It serves web content, which can include HTML pages, images, CSS files, JavaScript files, and other resources, in response to client requests. operate based on the Hypertext Transfer Protocol (HTTP) or its secure variant, HTTPS. When a client sends an HTTP request to the web server, specifying the desired resource (such as a webpage URL), the server processes the request and returns an HTTP response containing the requested**

**. What is REPL? Explain uses of REPL. REPL stand for Read, Eval, Print, Loop. It is a virtual environment where user inputs are read and evaluated and the results are returned to the user. Node.js comes with a REPL environment when it is installed, known as Node Interactive Shell. The REPL environment provides a convenient way to test JavaScript code. The terms in REPL are as follows: • Read: It reads the user inputs and parses it into JavaScript code and stored to the memory. • Eval: It evaluates (processes) the user inputs line by line. • Print: The results of the evaluation are printed to the console. • Loop: This Read-Eval-Print process is repeated for every line of code entered by the user. Uses of REPL: ¬ Experimenting: Node.js REPL environment can be used to test different ways to solve a problem. ¬ Debugging: When facing an error in JavaScript code, REPL helps to find the issue faster than running the code on a browser context. ¬ Learning: New methods and functions can be easily understood by first trying them on REPL environment. ¬ Quick Calculations: REPL can be used to make quick calculations necessary in a program without having to write the entire program. This helps in better program development. ( Draw The Diagram For REPL). Streams are the objects that facilitate reading data from a source and writing it to a destination.There are four types of streams in Node.js: • Readable: This stream is used for read operations. • Writable: This stream is used for write operations. • Duplex: This stream can be used for both read and write operations. • Transform: It is type of duplex stream where the output is computed according to input. Streams are a way to handle reading/writing file operations, network communications, or any kind of end-to-end information exchange in an efficient way. ▪ Instead of a program loading the entire file contents into memory at once, streams read chunks of data piece by piece, processing its content without keeping it all in memory.**

**1.Synchronous/Blocking Approach/Operations • Synchronous is a blocking architecture, so the execution of each operation depends on completing the one before it. • Synchronous execution means the code is executed in sequence line by line (one line at a time). For example, when a function is called, the program execution waits until that function returns before going to the next line of code. • Synchronous is blocking — it will only send the server one request at a time and wait for that request to be answered by the server. • Synchronous is slower and more methodical. • Eg. Traditional Web Server Model is synchronous and blocking 2.Asynchronous/Non-Blocking Approach/Operations • Asynchronous is a non-blocking architecture, so the execution of one task isn’t dependent on another. Tasks can run simultaneously. • Asynchronous execution means the code doesn’t necessarily run in the sequence. The program doesn’t wait for the code block to finish its execution and can move on to the next piece of code. • Asynchronous is non-blocking, which means multiple requests can be processed in parallel without waiting for the earlier request processing to be completed. • Asynchronous increases throughput because multiple operations can run at the same time. • Eg. Node.js Process Model is Asynchronous and Non-Blocking.What is the difference between exports and module.exports? \* Module is a plain Javascript object representing the current module.it is local to each module and it is also private. \* It has exports property which is plain Javascript variable that happens set to module.exports. \* When you require to export module in another file , the code within that module is executed , and only the module. exports in returned. \* Exports is just a reference to 'module.exports', and assigning a new**

**object to 'exports' breaks that reference as updating 'exports' no longer updates 'module.exports'.**

**function add(a, b) {**

**return a + b;**

**}**

**module.exports.add = add;**

**function add(a, b) {**

**return a + b;**

**}**

**exports.add = add;**

**If you try to access add or subtract on the exported object, an error is thrown because 'module.exports' is empty. \* So, while 'module.exports' and 'exports' may seem interchangeable, they are not the same.**

**What is NPM ? What are the advantages of NPM ? NPM is the default package manager for Node.js, a runtime environment for executing JavaScript code server-side. It is a CLI (command line interface)that helps to install, manage, and share packages or libraries of reusable JavaScript code.Advantages of using NPM in Node.js include:1. Package Management: NPM simplifies the process of installing, updating, and managing external libraries and dependencies in your Node.js projects.2. Dependency Resolution: It automatically handles the resolution of dependencies, ensuring that the correct versions of packages are installed.3. Version Control: NPM allows you to specify the versions of packages your project depends on, providing control over updates to avoid compatibility issues.4. Centralized Repository: Packages are stored in a centralized registry, making it easy to discover, share, and distribute Node.js modules.5. Script Execution: NPM allows you to define and run scripts, automating common tasks and making it easier to manage your project .6. Community and Ecosystem: Being a widely used package manager, NPM has a vast and active community, leading to a rich ecosystem of open-source packages and tools.**

**What is Module? What are the different types of modules? A module in Node.js is a self-contained unit of code that encapsulates specific functionality. It helps in organizing code into reusable and maintainable components. Node.js supports different types of modules, each serving a distinct purpose within an application: (write three points name first)**

**1. Core/Built-in Modules:- These are modules provided by Node.js itself.- Core modules offer essential functionalities required for various tasks, such as file system operations, networking, and managing streams. - Examples of core modules include 'fs' for file system operations, 'http' for creating HTTP servers, and 'path' for handling file paths. - Core modules are automatically loaded when the Node.js process starts and do not require installation via npm.**

**2. Local Modules: - Local modules are modules created within the application by the developer. - They are typically used to organize code into smaller, reusable components to improve maintainability and readability. - Local modules are stored in separate JavaScript files within the application's directory structure. - Developers can create their own modules to encapsulate specific functionalities and import them into other parts of the application as needed**

**.3. Third-Party Modules:- These modules are developed by third-party developers and made available for public use via npm (Node Package Manager). - Third-party modules expand the capabilities of Node.js by providing additional functionalities beyond what is available in core modules. - Developers can easily integrate third-party modules into their applications by installing them via npm. - Examples of popular third-party modules include Express.js for building web applications, lodash for utility functions, and Axios for making HTTP requests.**

**What is package.json? How is it used to manage third party dependencies in your application? `package.json` is a file used in Node.js projects to manage dependencies, scripts, and metadata about the project. It's typically located at the root directory of the project and is in JSON**

**format. \*Here's what `package.json` typically includes:**

**1. Project metadata : This includes information such as the project name, version, description, author, license, and other relevant metadata. 2. Dependencies : This section lists all the third-party libraries and packages that the project depends on to function properly. These dependencies can be modules available on the npm (Node Package Manager) registry or local file paths.3. Dependencies : Similar to regular dependencies, but these are only required for development purposes, such as testing frameworks, build tools, etc. These dependencies are not needed in the production environment. 4. \*\*Scripts\*\*: This section defines custom scripts that can be executed via npm. These scripts can be used for various purposes like running tests, building the project, starting a development server, etc.**

**`package.json` is used to manage third-party dependencies in the following ways:**

**1. \*\*Installation\*\*: When a project is initially set up or when dependencies are updated, developers can run `npm install` in the project directory. This command reads the `dependencies` and `devDependencies` sections of `package.json` and installs all the required packages.**

**2. \*\*Dependency Management\*\*: Developers can manually add, remove, or update dependencies in the `package.json` file. Once the changes are made, running `npm install` will update the project's dependencies accordingly.**

**3. \*\*Dependency Versioning\*\*: Each dependency in `package.json` typically specifies a version range or a specific version number. This allows developers to define the desired version of each dependency and ensures consistency across different environments.4. \*\*Sharing and Collaboration\*\*: `package.json` serves as a documentation of project dependencies, making it easier for developers to share their projects with others. By sharing the `package.json` file, other developers can quickly install all the required dependencies for the project.**

**Unit 2 //App.js(module)**

**const\_maskCardNumber=require("./”);const cardnumber = process.argv[2];console.log(maskCardNumber(No));**

**//another file**

**Function maskCardNumber(){**

**// logic(yourthought)**

**} module.exports = function name;**

**Unit 4 var http = require('http');&(url“)**

**http.createServer(function(req,res){res.writeHead(200, {'Content-Type' : 'text/html'});var q = url.parse(req.url, true).query;**

**var str = q.str;**

**//logic(Yourthought);**

**res.end(count);**

**}.listen(8080);**

**Unit 5**

**var http = require('http');&( “)**

**http.createServer(function(req,res){res.writeHead(200, {'Content-Type' : 'text/html'});**

**fs.readFile('text2.txt' , 'utf-8' , function(err,data){ if(err) throw err;**

**//logic(Yourthought);**

**res.end(count.toString());**

**}.listen(8080);**

var mysql = require('mysql2');

var con = mysql.createConnection({

    host: 'localhost',

    user: 'root',

    password: '',

*// database: 'actor'*

})

function createdatabase() {

    con.connect(function (err) {

        if (err) throw err;

        console.log("Connected");

    })

    con.query("create database actors", function (err) {

        if (err) throw err;

        console.log("database created!!!");

    }) }

createtable();

function createtable() {

    con.query("create table act(actID INT,actName varchar(255))", function (err, result, fields) {

        if (err)         throw err;

        console.log("Table Created...");

    }) }

*// insertData();*

function insertData() {

    var sql = "Insert into act values ? ";

    var values = [

        [101, 'shahrukh'],

        [102, 'akshay'],

        [103, 'salman'],

        [104, 'amir']

    ]

    con.query(sql, [values], function (err, result) {

        if (err) {

            throw err;

        }

        console.log("data inserted");

        console.log(result);

    })

}