**1) What is JavaScript?**

JavaScript is a text-based programming language used both on the client-side and server-side that allows you to make web pages interactive.

a) Text based programming language.

b) Used on Client side and server side.

c) Makes web pages interactive.

### 2) What is the difference between Java & JavaScript?

|  |  |
| --- | --- |
| **Java** | **JavaScript** |
| Java is an OOP programming language. | JavaScript is an OOP scripting language. |
| It creates applications that run  in a virtual machine or browser. | The code is run on a browser or node  Platform. |
| Java code needs to be compiled. | JavaScript code are all in the  form of text. |

### 3) ****What are the data types supported by JavaScript?****

### The data types supported by JavaScript are: Undefined, Null, Boolean, String, Symbol, Number, Object

### 4) ****What are the features of JavaScript?****

### **The primary features of Javascript are: it is Lightweight & it is an Interpreted programming language which is Good for the applications which are network-centric. On top of this as the language uses most of the keywords in JAVA it is complimentry to java language which most of the developers are familiar with. Javascript blends well with HTML as it uses the DOM tree to intrepret HTML for us to play around with elements.**

### 5) Is JavaScript a case-sensitive language?

### Yes, JavaScript is a case sensitive language. Here keywords, variable names, function names, and any other identifiers are to be used in the program consistentently and carefully.

### 6) List some advantages of javaScript?

### Few of the advantages in javascript are:

### In JS Server interaction is very less.

### The Feedback to the visitors is immediate.

### Interactivity is high as we can toggle the inputs inside browser developer tools and get an instant output.

### 7) Difference between “ == “ and “ === “ operators.

Both are comparison operators. The difference between both the operators is that “==” is used to compare values whereas, “ === “ is used to compare both values and types on LHS & RHS.

**8) Explain Hoisting in javascript.**

Hoisting is the default behaviour of javascript where all the variable and function declarations are moved on top.

Eg: function m1(){ var a;

for(var i; i <=10; i++){

console.log(i);

}

console.log(i); //Variable access is permitted because it is hoisted

}

**9) What is the purpose of ‘this’ keyword in JavaScript?**

The JavaScript this keyword refers to the object it belongs to. This has different values depending on where it is used. In a method, this refers to the owner object and in a function, this refers to the global object.

**10) How to create an array in JavaScript?**

There are 3 ways to create an array in JavaScript.

* By using array literal.
* By creating an instance of Array
* By using an Array constructor

Let's see a simple code to create an array using object literal.

var emp=["Shyam","Vimal","Ratan"];

**11) Is javascript a statically typed or a dynamically typed language?**

JavaScript is a dynamically typed language. In a dynamically typed language, the type of a variable is checked during run-time in contrast to a statically typed language, where the type of a variable is checked during compile-time.

For example, a variable that is assigned a number type can be converted to a string type:

var a = 23;

var a = "Hello World!";

**12) What is NaN property in JavaScript?**

NaN property represents the “Not-a-Number” value. The typeof of NaN will return a Number.

To check if a value is NaN, we use the isNaN() function,Eg:

* isNaN("Hello") // Returns true
* isNaN(345) // Returns false
* isNaN('1') // Returns false, since '1' is converted to Number type which results in 0 ( a number)
* isNaN(true) // Returns false, since true converted to Number type results in 1 ( a number)
* isNaN(undefined) // Returns true

**13) Explain Higher Order Functions in javascript.**

Functions that operate on other functions, either by taking them as arguments or by returning them, are called higher-order functions.

Example:

function higherOrder(fn) {

fn();

}

higherOrder(function() { console.log("Hello world") });

**14) In JavaScript what is an argument object?**

The variables of JavaScript represent the arguments that are passed to a function.

**15)** **What is the difference between undeclared & undefined variables?** The difference between undeclared and undefined variables are: An undeclared variable has not been declared anywhere in the code, so said variable does not exist. If you try to read an undeclared variable, JavaScript throws an error. An undefined variable has been declared in the program, but no value has been assigned. This means the variable exists, but its value is yet to be defined.

**16) How to write HTML code dynamically using JavaScript?** The innerHTML property is used to write the HTML code using JavaScript dynamically.

Eg: document.getElementById('mylocation').innerHTML="<h2>This is heading using JavaScript</h2>";

**17) What are Constructors in JavaScript?**

Constructor in JS are the most powerful members of a class which will be used for object initialization. The special ability of a aconstructor is, it will be invoked only once during the creation of an object.

Eg: function User(name) {

this.name = name;

}

let firstUser = new User('John Doe');

**18)** **How Can You Return a Character From a Specified Index?**

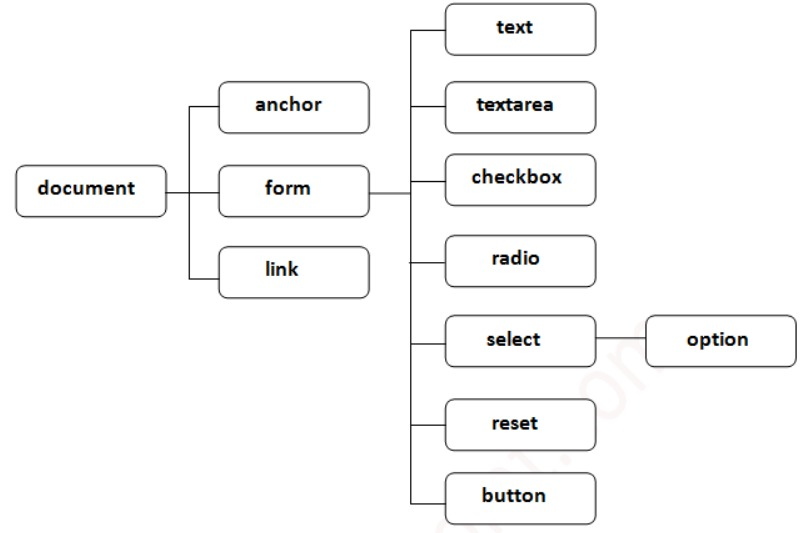
By using the charAt() method, we can retrieve the first char at index 0 to fetch J. As seen in below example

Eg: let userName = "John";

console.log(userName.charAt(0)); // "J"

**19) What is DOM ? (coorection)**

The Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document.



**20) What are the arrow functions in JavaScript?**

Arrow functions are a short and concise way of writing functions in JavaScript. It was introduced in ES6 specification. The general syntax of an arrow function is as below:

const helloWorld = () => {

console.log("hello world!");

};

**21)What is a specification?**

A) Specfication are the programming templates introduced by the Consortium( check spelling w3C) in collaboration & discussion with various vendors to implement the templates: for the developers to follow/use in their programs.

Eg: w3c sets the standards & gives the tag names for HTML/CSS in discussion with various browser vendors. Different browser vendors will implement these tags, attributes & values with the same name as specified by the Consortium. Developers will use these tags, attribute & values in their programming.

**22)How do you create an object in JavaScript?**

We can create objects in JavaScript using two ways. One is Instant object creation by mentioning properties inside flower braces within a program.

Other one is by creating a class template & later create object of that template by using the “new” keyword.

Example: const student = {

name: 'John',

age: 17

}

**23)What are the conventions of naming a variable in JavaScript?**

Following are the naming conventions for a variable in JavaScript:

* Variable names cannot be same as reserved keywords. For example, var, let, const, etc.
* Variable names cannot begin with a numeric value. They must only begin with a letter or an underscore character.
* Variable names are case-sensitive.

**24)What is Callback in JavaScript?**

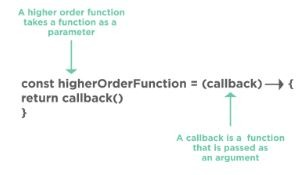
In JavaScript, functions are objects and therefore, functions can take other functions as arguments and can also be returned by other functions.

Fig: Callback function.

A callback is a JavaScript function that is passed to another function as an argument or a parameter. This function is to be executed whenever the function that it is passed gets executed.

**25)How do you debug a JavaScript code?**

All modern web browsers like Chrome, Firefox, etc. have an inbuilt debugger that can be accessed anytime by pressing the relevant key, usually the F12 key. There are several features available to users in the debugging tools.

We can also debug a JavaScript code inside a code editor that we use to develop a JavaScript application—for example, Visual Studio Code, etc.

**26)What are the ways of adding JavaScript code in an HTML file?**

There are primarily two ways of embedding JavaScript code:

* We can write JavaScript code within the script tag in the same HTML file; this is suitable when we need just a few lines of scripting within a web page.
* We can import a JavaScript source file into an HTML document; this adds all scripting capabilities to a web page without cluttering the code.

**27)What do you understand about cookies?**

A cookie is generally a small data that is sent from a website and stored on the user’s machine by a web browser that will be used to access certain features in the designed website. Cookies are used to remember information for later use and also to record the browsing activity on a website.

**28)How would you create a cookie?**

The simplest way of creating a cookie using JavaScript is as below:

document.cookie = "key1 = value1; key2 = value2; expires = date";

**29)How would you read a cookie?**

Reading a cookie using JavaScript is also very simple. We can use the document.cookie string that contains the cookies that we just created using that string.

The document.cookie string keeps a list of name-value pairs separated by semicolons, where ‘name’ is the name of the cookie, and ‘value’ is its value. We can also use the split() method to break the cookie value into keys and values.

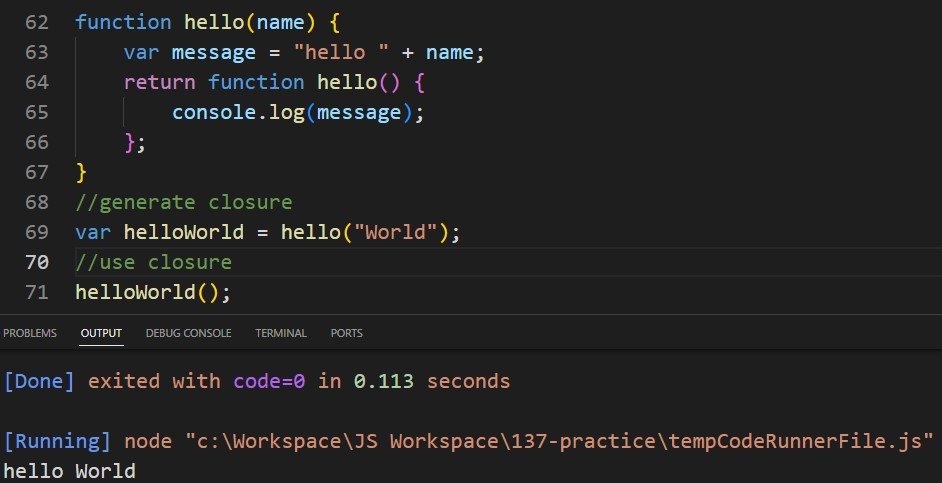
**30)What are the different ways an HTML element can be accessed in a JavaScript code?**

Here are the ways an HTML element can be accessed in a JavaScript code:

* getElementByClass(‘classname’): Gets all the HTML elements that have the specified classname.
* getElementById(‘idname’): Gets an HTML element by its ID name.
* getElementbyTagName(‘tagname’): Gets all the HTML elements that have the specified tagname.
* querySelector(): Takes CSS style selector and returns the first selected HTML element.

**31)What are Closures in JavaScript?**

Closures provide a better, and concise way of writing JavaScript code for the developers and programmers. Closures are created whenever a variable that is defined outside the current scope is accessed within the current scope.

Example: 

**32)What is the Session storage and Local storage?**

Session Storage: The data stored in session storage gets expired or deleted when a page session ends.

Local Storage: Websites store some data in local machine to reduce loading time; this data does not get deleted at the end of a browsing session.

**33)The difference between exec() and test() methods (Question realted to regax)**

exec(): It is an expression method in JavaScript that is used to search a string with a specific pattern.

Once it has been found, the pattern will be returned directly, otherwise, it returns an “empty” result.

test() : It is an expression method in JavaScript that is also used to search a string with a specific pattern or text.

Once it has been found, the pattern will return the Boolean value 'true', else it returns ‘false’.

**34)What are object prototypes?**

Whenever we create a function using JavaScript, JavaScript engine will automatically attach with hidden properties and functions. This hidden properties & methods come from prototype. The same phenomenon is valid for creating arrays & other objects.

**35)What is BOM?**

BOM is the Browser Object Model where users can interact with browsers that is a window, an initial object of the browser. The window object consists of a document, history, screen, navigator, location, and other attributes. Nevertheless, the window’s function can be called directly as well as by referencing the window.

**36)Types of errors in JavaScript**

JavaScript types of errors are,

Load time errors: Errors that come up when loading a web page, like improper syntax errors, are known as Load time errors and generate the errors dynamically.

Runtime errors: Errors that come due to misuse of the command inside the HTML language.

Logical errors: These are the errors that occur due to the bad logic performed on a function with a different operation.

**37)Difference between client-side and server-side.**

* Client-side JavaScript

Client-side JavaScript is made up of fundamental language and predefined objects that perform JavaScript in a browser.

Also, it is automatically included in the HTML pages where the browser understands the script.

* Server-side JavaScript

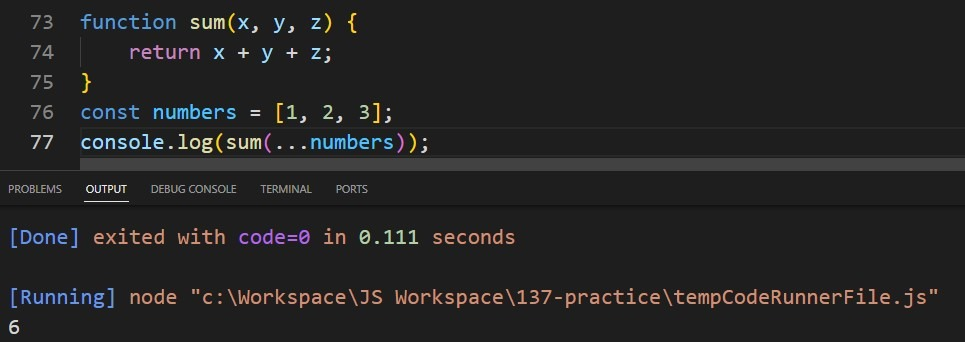
Server-side JavaScript is quite similar to Client-side JavaScript.

Server-side JavaScript can be executed on a server. Eg: Node JS

The server-side JavaScript is deployed once the server processing is done.

**38)Spread Operator(...). (Correction)**

It is an operater used to copy all properties in an object into a new object. Eg:



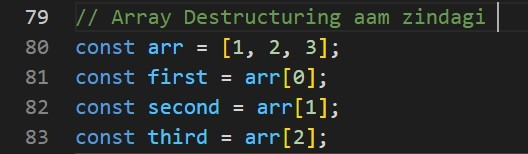
**39)What are generator functions?**

Generator functions are declared with a special class of functions and keywords using function\*. It does not execute the code, however, it returns a generator object and handles the execution.

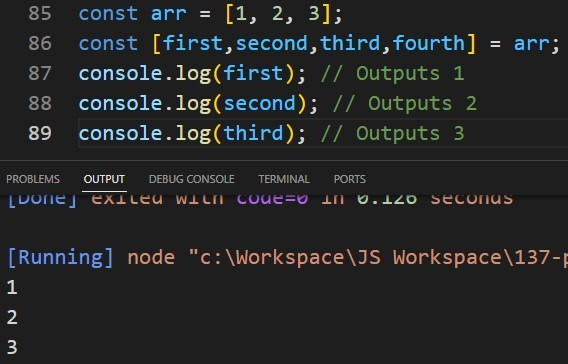
**40)What is Object Destructuring? (with examples)**

Object destructuring is a method to extract elements from an array or an object.

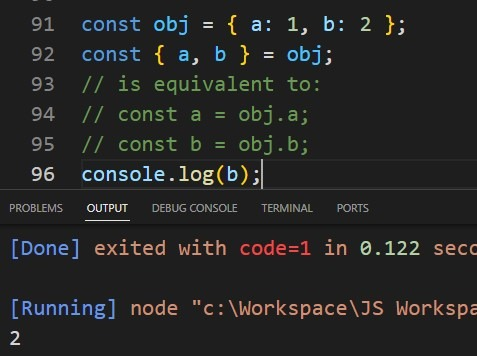
Example 1:



Example 2:



**Example 3:**



**41)Difference between Async/Await and Generators .**

* Async/Await

Async-await functions are executed sequentially one after another in an easier way.

Async/Await function might throw an error when the value is returned.

* Generators

Generator functions are executed with one output at a time by the generator’s yield by yield.

The ‘value: X, done: Boolean’ is the output result of the Generator function. I have not worked on generators

**42)How do you empty an array in JavaScript?**

There are a few ways in which we can empty an array in JavaScript:

* By assigning array length to 0:

var arr = [1, 2, 3, 4];

arr.length = 0;

* By assigning an empty array:

var arr = [1, 2, 3, 4];

arr = [];

* By popping the elements of the array:

var arr = [1, 2, 3, 4];

while (arr.length > 0) {

arr.pop();

}

* By using the splice array function:

var arr = [1, 2, 3, 4];

arr.splice(0, arr.length);

**43)What is a prompt box?**

A prompt box is a box that allows the user to enter input by providing a text box. A label and box will be provided to enter the text or number.

**44)How can you convert the string of any base to an integer in JavaScript?**

The parseInt() function is used to convert numbers between different bases. parseInt() takes the string to be converted as its first parameter. The second parameter is the base of the given string.

Example: parseInt ("4F", 16);

**45)What would be the result of 3+2+”7″?**

Since 3 and 2 are integers, they will be added numerically. And since 7 is a string, its concatenation will be done. So the result would be 57.

**46)What are all the types of Pop up boxes available in JavaScript?**

* Alert
* Confirm and
* Prompt

**47)What is the difference between an alert box and a confirmation box?**

An alert box displays only one button, which is the OK button.

But a Confirmation box displays two buttons, namely OK and cancel.

**48)What are escape characters?**

Escape characters (Backslash) is used when working with special characters in our javascript program like single quotes, double quotes, apostrophes, and ampersands. Place backslash before the characters to make it accept as a valid string input..

Example:

document. write "I m a "good" boy."

document. write "I m a \"good\" boy."

**49)What a pop()method in JavaScript is?**

The pop() method is similar to the shift() method, but the difference is that the Shift method works at the array’s start. The pop() method takes the last element off of the given array and returns it. The array on which it is called is then altered.

Example:

var cloths = ["Shirt", "Pant", "TShirt"];

cloths.pop();

//Now cloth becomes Shirt,Pant

**50)What is break and continue statements?**

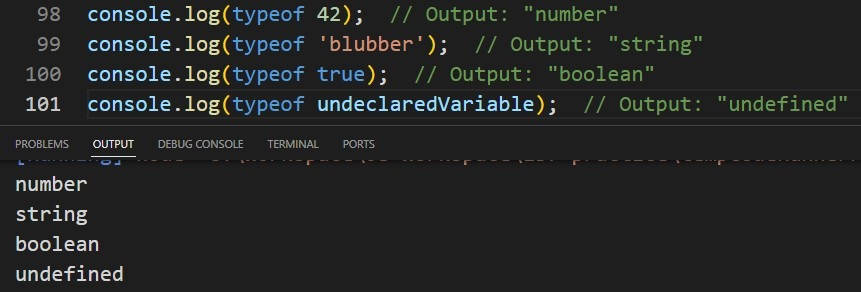
Break statement exits from the current loop.

Continue statement continues with next iteration of the loop & skips the below statements.

**51)What is the use of a type of operator?**

‘Typeof’ is an operator used to return a string description of the type of a variable.

Example:



**52)Which keywords are used to handle exceptions?**

Try… Catch—finally is used to handle exceptions in the JavaScript

Example:

try{

Code

}catch(exp){

Code to throw an exception.

}finally{

Code runs either it finishes successfully or after catch

}

**53) What are some of the built-in methods in JavaScript?**

* Date()-Returns the present date and time.
* concat()-Joins two strings and returns the new string.
* push()-Adds an item to an array.
* pop()-Removes and also returns the last element of an array.
* round()-Rounds of the value to the nearest integer and then returns it.
* length()-SReturns the length of a string.

**54)What are Imports and Exports in JavaScript?**

Imports and exports help in writing modular code for our JavaScript applications. With the help of imports and exports, we can split a JavaScript code into multiple files in a project. This greatly simplifies the application source code and encourages code readability.

calc.js

export const sqrt = Math.sqrt;

export function square(x) {

return x \* x;

}

export function diag(x, y) {

return sqrt(square(x) + square(y));

}

This file exports two functions that calculate the squares and diagonal of the input respectively.

main.js

import { square, diag } from "calc";

console.log(square(4)); // 16

console.log(diag(4, 3)); // 5

Therefore, here we import those functions and pass input to those functions to calculate square and diagonal.

Keywords import & export are Java Inspired & are only applicable for Browzer based ES6 Run time envoronment. & not Node based environent.

For Node based we use require() & module.exports inspired from Ruby.

**55) What are some of the JavaScript frameworks and their uses?**

JavaScript has a collection of many frameworks that aim towards fulfilling the different aspects of the web application development process. Some of the prominent frameworks are:

React - Frontend development of a web application.

Angular - Frontend development of a web application.

Node - Backend or server-side development of a web application.

Express

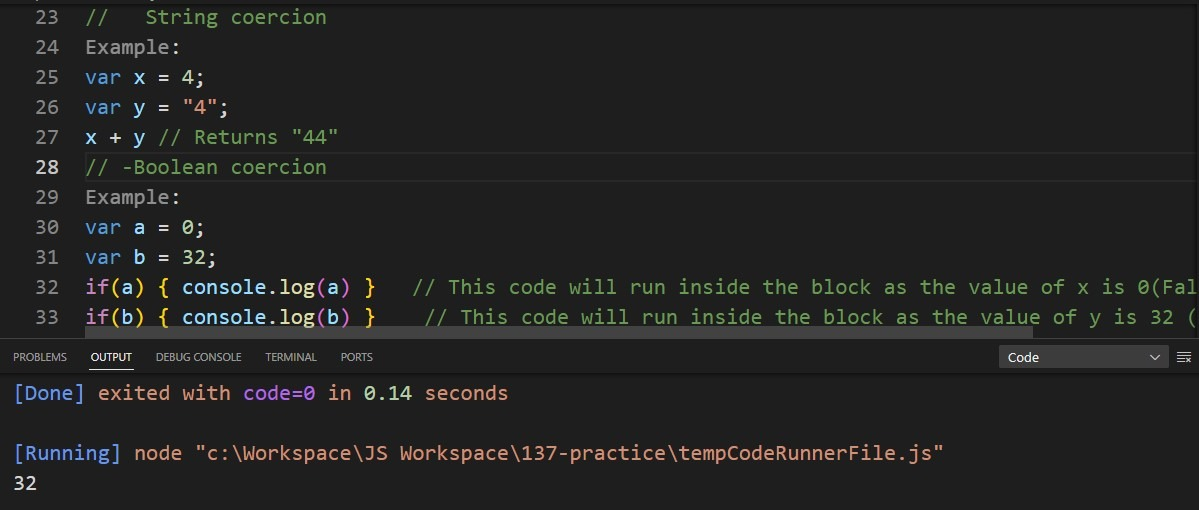
EJS

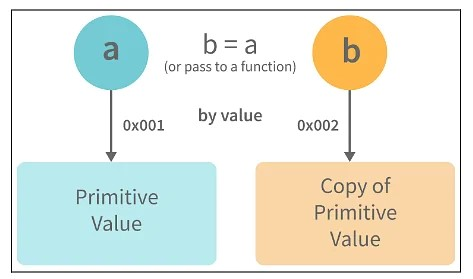
**56) What is the difference between Call and Apply? (explain in detail with examples)**

* call() method is using ‘pass by reference’ as we pass the object on which the method is to be called (with some additional parameters: comma separated)
* apply() method is using ‘pass by reference’ as we pass the object on which the method is to be called (with some additional parameters: as an array)
* bind() method returns a Function, but binds the function to the object & arguments we are passing (and keeps it ready). To invoke it, we have to exclusively use the call() method (if of the same line) -or- invoke it by using a parentheses in the consecutive line, by accepting the return of the function object (returned by the bind method)

**57) Implicit Type Coercion in javascript (in detail with examples)**

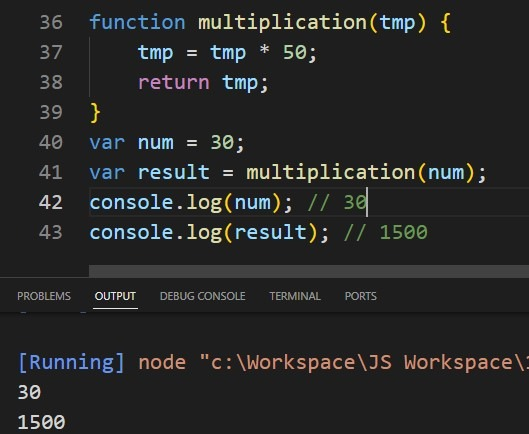
When the value of one data type is automatically converted into another data type, it is called Implicit type coercion in javascript.



**58) What is Pass by value and pass by reference**

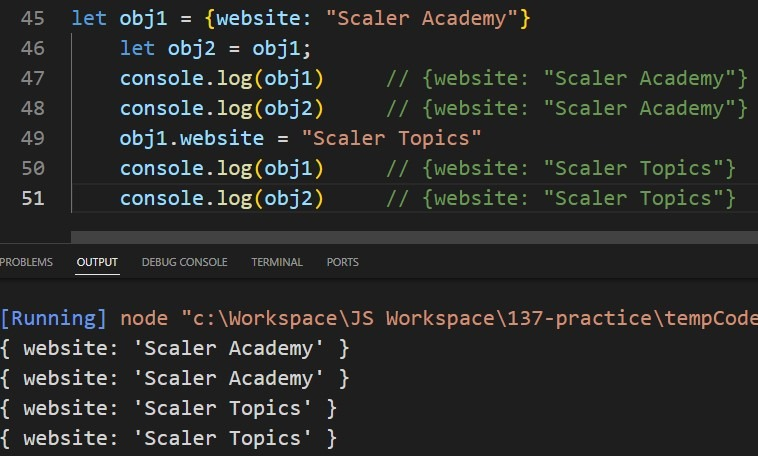
In Pass by Value, Function is called by directly passing the value of the variable as the argument. Changing the argument inside the function doesn’t affect the variable passed from outside the function.

Example:



In Pass by Reference, Function is called by directly passing the reference/address of the variable as the argument. Changing the argument inside the function affect the variable passed from outside the function. In Javascript objects and arrays follows pass by reference.

Example:



**59) Self Invoking Functions**

Self Invoking Functions is an automatically invoked function expression followed by (), where it does not need to be requested. Nevertheless, the declaration of the function is not able to be invoked by itself.

Example: (function () {

let x = "Hello!!"; // I will invoke myself

})(); //output:Hello

**60) Advantages of using External JavaScript**

External Javascript allows web designers and developers to collaborate on HTML and javascript files.

It also enables you to reuse the code.

External javascript makes Code readability simple.

**61) Can you assign an anonymous function to a variable and pass it as an argument to another function?**

Yes! An anonymous function can be assigned to a variable. It can also be passed as an argument to another function.

Example: var greet = function () {

console.log("Welcome to World!");

};

greet();

**62) What are the ways to define a variable in JavaScript?**

The three possible ways of defining a variable in JavaScript are:

(i) var – var is the old keyword of JavaScript which will hoist the variable at the function level.

Example: var a =10; Variable declarations are processed before the execution of the code.

(ii) const – const is a keyword used to declare a variable which stores a constant value throughout the scope and it cannot be hoisted.

(iii) let – let is a keyword used to declare a variable within a specified scope and kills the variable after the scope ends.

**63) What is the difference between null & undefined?**

Undefined means a variable has been declared but has not yet been assigned a value. On the other hand, null is an assignment value. It can be assigned to a variable as a representation of no value. Also, undefined and null are two distinct types: undefined is a data-type by itself (undefined) while null is an object.

**64) What is the difference between innerHTML & innerText?**

@ innerHTML – It will process an HTML tag if found in a string

Example: document.getElementsByTagName('body')[0].innerHTML = '<div id="gator">Alligators rule!!</div>';

@ innerText – It will not process an HTML tag if found in a string

Example: document.getElementById('gator').innerText = 'OF COURSE alligators rule!';

**65) How to write a comment in JavaScript?**

There are two types of comments in JavaScript.

Single Line Comment: It is represented by // (double forward slash)

Multi-Line Comment: Slash represents it with asterisk symbol as /\* write comment here \*/

**66) What are the falsy values in JavaScript, and how can we check if a value is falsy?**

Those values which become false while converting to Boolean are called falsy values. when used inside a relational expresson turns into false

const falsyValues = ['', 0, null, undefined, NaN, false];

**67) What is the use of Math object in JavaScript?**

The JavaScript Math object provides several constants and methods to perform a mathematical operation. Unlike date object, it doesn't have constructors. For example:

function display()

{

document.writeln(Math.random());

}

display();

**68) What is the use of a Date object in JavaScript?**

The JavaScript date object can be used to get a year, month and day. You can display a timer on the webpage by the help of JavaScript date object.

function display()

{

var date=new Date();

var day=date.getDate();

var month=date.getMonth()+1;

var year=date.getFullYear();

document.write("<br>Date is: "+day+"/"+month+"/"+year);

}

display();

**69) What is the use of a Number object in JavaScript?**

The JavaScript number object enables you to represent a numeric value. It may be integer or floating-point. JavaScript number object follows the IEEE standard to represent the floating-point numbers.

function display()

{

var x=102;//integer value

var y=102.7;//floating point value

var z=13e4;//exponent value, output: 130000

var n=new Number(16);//integer value by number object

document.write(x+" "+y+" "+z+" "+n);

}

display();

**70) What is the use of a Boolean object in JavaScript?**

The JavaScript Boolean is an object that represents value in two states: true or false. You can create the JavaScript Boolean object by Boolean() constructor.

function display()

{

document.writeln(10<20);//true

document.writeln(10<5);//false

}

display();

**71) What is an identifier?**

An identifier is the name given to a variable or a function or a class in a javascript program.

**72) What are the rules while declaring JavaScript Variable?**

\* A name should start with letters (a-z & A-Z) or underscore(\_) or $ dollar sign.

\* It should not start with special characters and numbers.

\* In JavaScript variables are case sensitive.

\* Reserved Keywords cannot be used as names in JavaScript.

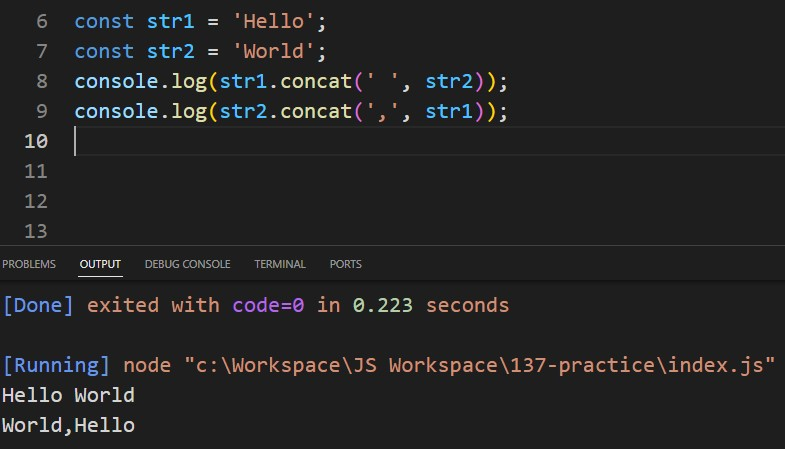
**73) What is string in JavaScript?**

String in JavaScript is an object that represents sequence of characters inside ' ' or " ".

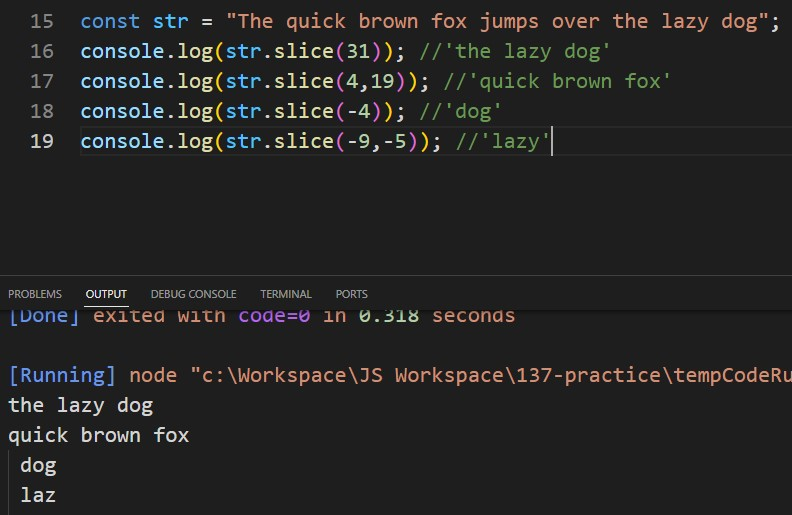
**74) What is concat method?**

The concat() method concatinates the string argument to the calling string and returns the new string.

Eg:

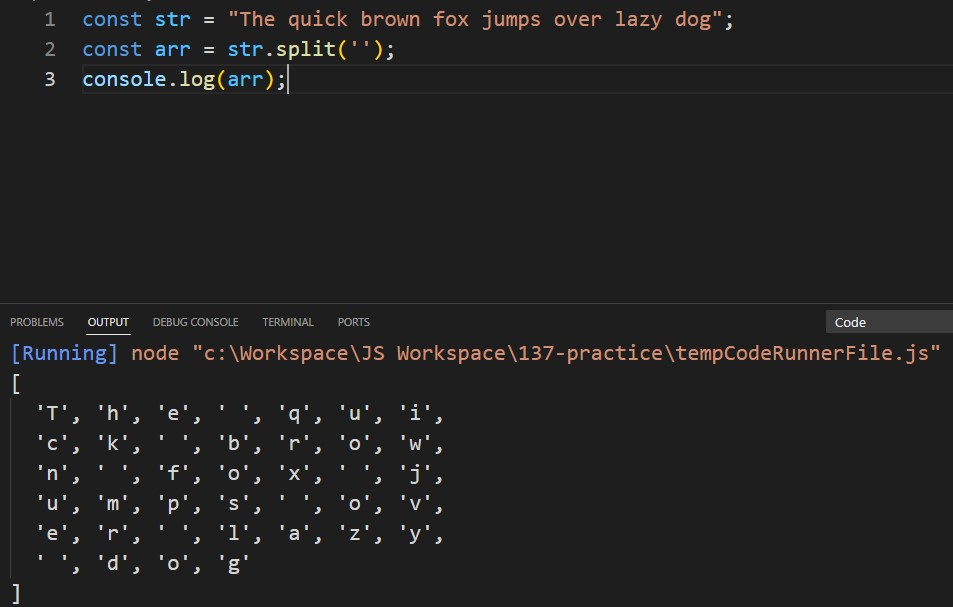


**75) Explain slice() method.**

The slice() method extracts a section of a string and returns it as new string.

**76) Explain split() method.**

The method takes a pattern input and divides the original string into an array of substring. The following returns an array as an output. The pattern here is know as "delimiter".



**77) Explain JavaScript Template literals ?**

Template literals is a string in JavaScript enclosed by backticks(`) allowing the programmer to perform string interpolation.

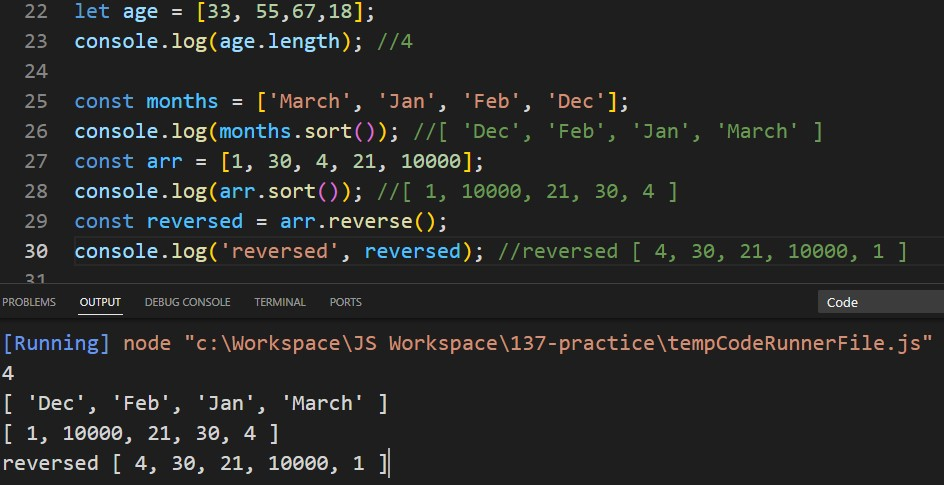
**78) What are the methods used in Array?**

1.Length: The length property inside an array holds the real-time length of the object. We can access the length property using (.) dot operator.

2.sort(): The sort method inside an array object will sort the elements inside. It will also return new sorted array reference but there is no need to catch the reference as previous reference will also work fine.

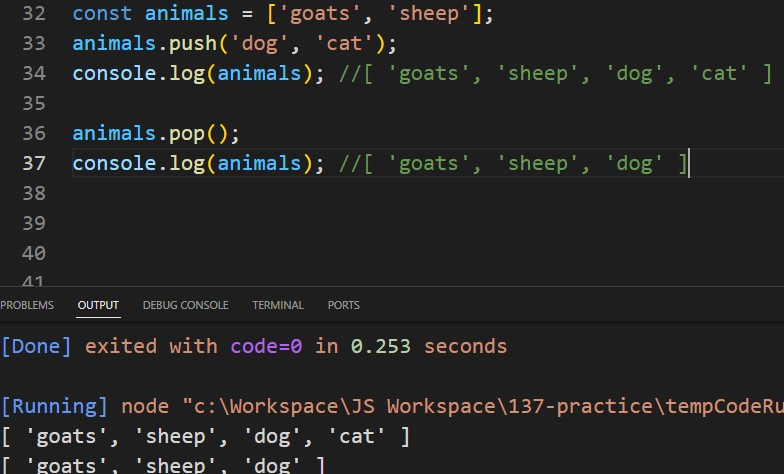
3. reverse(): The reverse method reverses the content inside an array and returns the references to the same array.

Example:



4.push(): The push method adds the specified elements to the end of an array and returns only the new length values.

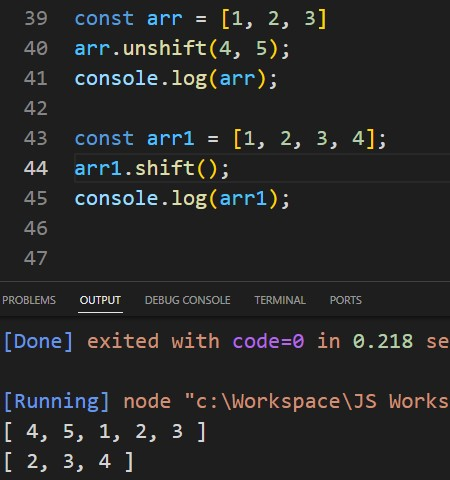
5.pop(): The pop method removes the last element from an array and returns the element itself.

Example:

6.unshift(): The unshift method adds the specified element to the beginning of an array and returns the new length of the array.

7.shift(): The shift method removes the first element from an array and returns that removed element. This method changes the length of the array.

Example:



**79) What is Relational operators?**

The relational operators allows us to compare two values to see whether they are equal,unequal greater than or less than.

**80) What is search() method?**

The search() method executes a search for a match between a regular expression and the current string object.

**81) What is a regular expression?**

A regular expression is a sequence of characters that forms a search pattern. It is used to find patterns in strings or sets of strings.

Example: let text = "Visit W3Schools";

let pattern = /w3schools/i;

let result = text.match(pattern); //W3Schools

**82) What’s the difference between greedy and nongreedy matching?**

In regular expression matching, greedy matching means that the regular expression engine will try to match as much of the string as possible. Non-greedy matching means that the engine will try to match as little of the string as possible.

**83) What are quantifiers used for in regular expressions?**

Quantifiers are used to specify how many times a character, group, or element can occur in a given regular expression. For example, the quantifier “\*” means that the character, group, or element can occur zero or more times, while the quantifier “+” means that the character, group, or element must occur at least once.

Example: var str = "Hello, welcome to edureka! 1 12 123";

var quant1 = /e+/g;

var quant2 = /el\*/g;

var quant3 = /1?/g;

var quant4 = /d{2}/g;

Output: e,e,e,e,e

ell,el,e,e,e

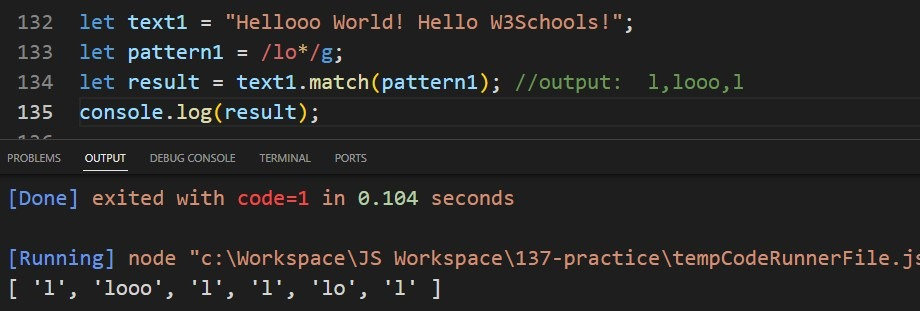
,,,,,,,,,,,,,,,,,,,,,,,,,,,1,,1,,,1,,,

12,12

**84) What does the \* operator do in a regular expression?**

The \* operator is a quantifier, which indicates that the preceding element can be matched zero or more times.

Example:



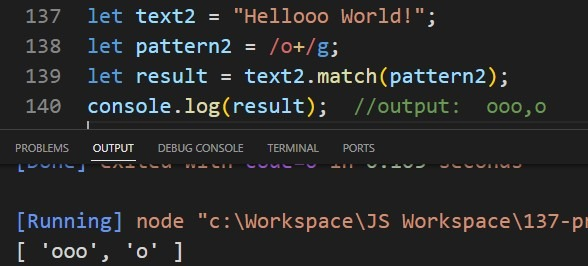
**85) Which special character is used to find any single character that matches a pattern?**

The period (.) is used to find any single character that matches a pattern.

**86) What does the + operator do in a regular expression?**

The + operator is used to indicate one or more occurrences of the preceding element.

Example:



**87) What does the ? operator do in a regular expression?**

The ? operator is a quantifier that specifies how many times an element can occur in a given pattern. For example, the regular expression a? specifies that an “a” can occur zero or one time in a given pattern.

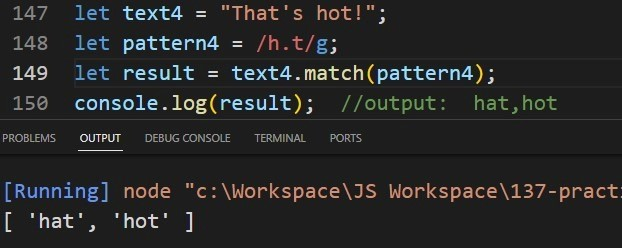
Example:



**88) What does the . operator do in a regular expression?**

The . operator is a wildcard character that matches any character except for a newline character.

Example:



**89) What do the ^ and $ operators do in a regular expression?**

The ^ operator matches the beginning of a string, while the $ operator matches the end of a string. These operators are often used together to match an entire string, from start to finish.

Example: ^a Search all string which start with "a".

a$ Search all string which end with "a".

Example2:



**90) What do the [] square brackets do in a regular expression?**

The [] square brackets are used to create a character class, which is a set of characters that can be matched by a single character.

For example, the character class [abc] will match any of the characters a, b, or c.

Example2: var str = "Edureka Online 123";

var ex1 = /[e]/gi;&nbsp; &nbsp; //[abc]

var ex2 = /[^e]/gi;&nbsp; &nbsp;//[^abc]

var ex3 = /[2]/g;&nbsp; &nbsp; &nbsp; //[0-9]

var ex4 = /[^2]/g;&nbsp; &nbsp; &nbsp;//[^0-9]

Output: E,e,e

d,u,r,k,a,O,n,l,i,n,1,2,3

2

E,d,u,r,e,k,a,O,n,l,i,n,e,1,3

**91) What do the {} curly braces do in a regular expression?**

The curly braces are used to denote a quantifier in a regular expression. This means that they specify how many times a character, group, or element can be repeated in order for the expression to match.

For example, the expression /a{2}/ would match any string that contains the letter ‘a’ twice in a row, such as “aba” or “aaa”.

**92) Can you give me some examples of how you would use regular expressions to validate data?**

There are many ways to use regular expressions to validate data. Some common examples include checking for a valid email address, phone number, or credit card number. You can also use regular expressions to ensure that a string contains only certain characters, or to check that a string is of a certain length.

**93) How would you write a Regular Expression to validate an email address pattern?**

A regular expression to validate an email address pattern can be written as follows:

Example: /^([a-zA-Z0-9.\_%-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6})\*$/

This regex checks for a sequence of alphanumeric characters (including .\_%-) followed by the @ symbol. After this, it looks for another sequence of alphanumeric characters (including . and -), then a period, and finally 2 to 6 alphabetic characters. The entire pattern is enclosed within start (^) and end ($) symbols to ensure that the whole string matches the pattern.

**94) How can you use Regular Expressions to split a string into an array?**

Regular Expressions (regex) can be used to split a string into an array by using the split() method in JavaScript. This method splits a String object into an array of strings by separating the string into substrates.

Example: let str = "Hello World! How are you today?";

let regex = /\s+/; // Matches one or more whitespace characters

let result = str.split(regex);

console.log(result); // Output: ["Hello", "World!", "How", "are", "you", "today?"]

**95) Can you write a Regular Expression to validate a URL?**

Yes, a regular expression can be used to validate a URL. Here’s an example:

Example: ^(https?:\/\/)?([\da-z\.-]+)\.([a-z\.]{2,6})([\/\w \.-])\/?$

This regex works as follows:

* ^ asserts start of line.
* (https?:\/\/)? matches the protocol (http or https), followed by ://. The question mark makes it optional.
* ([\da-z\.-]+) matches one or more digit, lowercase letter, dot or hyphen.
* \. matches a literal dot.
* ([a-z\.]{2,6}) matches between 2 and 6 of a lowercase letter or a dot.
* ([\/\w \.-]\*) matches zero or more of a forward slash, word character, whitespace, dot or hyphen.
* \/? optionally matches a trailing slash.
* $ asserts end of line.

**96) How do you handle case sensitive and insensitive matches in Regular Expressions?**

Regular expressions handle case sensitivity through the use of flags.

For case sensitive matches, no flag is needed as it’s the default behavior. For example, /abc/ will only match “abc”.

For case insensitive matches, the ‘i’ flag is used. This makes the regular expression ignore case. For instance, /abc/i will match “abc”, “ABC”, “AbC”, etc.

**97) Can you explain how you’d use the pipe ‘|’ symbol in a Regular Expression?**

The pipe ‘|’ symbol in a Regular Expression (regex) is used as a logical OR operator. It matches the pattern before or after it. For instance, ‘a|b’ will match either ‘a’ or ‘b’.

**98) What is the difference between ‘\d’, ‘\w’, and ‘\s’ in Regular Expressions?**

‘\d’, ‘\w’, and ‘\s’ are metacharacters in Regular Expressions, each with a unique function. ‘\d’ matches any digit from 0-9. It’s equivalent to [0-9]. On the other hand, ‘\w’ matches any alphanumeric character including underscore (). It is synonymous with [a-zA-Z0-9]. Lastly, ‘\s’ matches any whitespace character: spaces, tabs, line breaks etc. These metacharacters simplify pattern matching by representing common groups of characters.

**99) What is regular expression flags ?**

Regular expression flags sometimes called as modifiers , they are effect the way of pattren matching.

Different kinds of flags available: /g, /i etc.

**100) Is regular expression different in all languages?**

The syntax may vary slightly in other languages but has the same concept. The specific meaning of the metacharacters may also vary.

**101) What is a promise?**

A promises in JavaScript are used o handle asynchronous operations. A promise is an object that keeps track about whether a certain events happened already or not. It also determines what will happen after an event.

**102) What are the different states that a promise can be in during its lifetime?**

A promise can be in one of three states during its lifetime: pending, fulfilled, or rejected. A promise starts in the pending state, and then either transitions to the fulfilled state if the promise is successfully completed, or the rejected state if the promise is not completed.

**103) Is it possible to chain promises together? If yes, then how ?**

Yes, it is possible to chain promises together. This can be done by returning a promise from the then() function of a previous promise. The returned promise will then be resolved or rejected based on the resolution of the promise that it is chained to.

**104) Can you explain what the .then() method does?**

The .then() method is used to specify what should happen when a promise is resolved. This is usually used to chain together multiple promises, so that each promise is executed in order. The .then() method can take one argument followed by .catch() method: a success callback and a failure callback. The .then() method will be executed if the promise is resolved, and the .catch() method will be executed if the promise is rejected.

**105) Can you explain what the .catch() method does?**

The .catch() method is used to handle errors that may occur in a Promise chain. The .catch() method takes a single argument, which is a function that will be executed if an error occurs. The .catch() method is typically used after a .then() method to handle any errors that may have occurred in the .then() method.

**106) Why should we avoid using callback functions?**

There are a few reasons why callback functions should be avoided:

1.They can be hard to read and debug

2.They can lead to what is known as “callback hell”

3.Promises provide a simpler and more elegant way to handle asynchronous code

**107) What is prototype?**

In JavaScript, each object contains an internal hidden property [[Prototype]], which corresponds to the prototype of that object, which may be null or point to another object. However, since [[Prototype]] is an internal property that cannot be accessed directly, the browser provides the \_\_proto\_\_ access method, which can be referenced in the following code.

But we need to pay attention that the \_\_proto\_\_ method is not in the ECMAScript specification. In fact, when developing, we will use Object.getPrototypeOf to get the prototype of the object.

Example:function Person() {} // Person is a constructor function

const personA = new Person(); // Create a personA object through the Person constructor function

console.log(personA.\_\_proto\_\_); //output:{constructor:ƒ}//Through the proto method, view the prototype of personA

personA.\_\_proto\_\_ === Person.prototype; //output: true// personA object can be accessed to its prototype through the proto method

Object.getPrototypeOf(personA) === Person.prototype; // true

personA.\_\_proto\_\_ === Object.getPrototypeOf(personA); // true

**108) What is the difference between \_\_proto\_\_ and [[Prototype]] ?**

[[Prototype]] is a special hidden property in JavaScript objects, but because it cannot be directly accessed, it can be accessed through the proto access method.

**109) What is the difference between \_\_proto\_\_ and prototype?**

proto and prototype are different properties. proto is a hidden property of each object, and each object can access its prototype through proto. And prototype is a property that exists in all constructor functions, and the prototype of the constructor function is actually the same as the \_\_proto\_\_, which is called the prototype object.

Example: function Person() {} // Person is a constructor function

const personA = new Person(); // Create a personA object through the Person constructor function

personA.\_\_proto\_\_ === Person.prototype; // true

**110) What is the \_\_prototype\_\_ chain?**

Prototype is a special hidden property in JavaScript objects, and each object can access its prototype through proto. The prototype itself is an object, so it also has its own prototype. When we try to access an attribute of an object, if the object does not have the required attribute, it will look for it in its prototype. If the prototype still does not find it, it will continue to look up one level until it is found, or until it reaches null. This continuous path is called the prototype chain, and the end of the chain is null.

Example: personA.\_\_proto\_\_.\_\_proto\_\_.\_\_proto\_\_ === null;

**111) What is class inheritance in JavaScript?**

Class inheritance in JavaScript allows a subclass to inherit properties and methods from a parent class, enabling code reuse and reducing redundancy.

Example: class Animal {

constructor(name) {

this.name = name;

}

speak() {

console.log(`${this.name} makes a sound.`);

}

}

class Dog extends Animal {

constructor(name, breed) {

super(name);

this.breed = breed;

}

speak() {

console.log(`${this.name} barks!`);

}

fetch() {

console.log(`${this.name} fetches the ball.`);

}

}

const myDog = new Dog("Buddy", "Golden Retriever");

myDog.speak(); // Output: "Buddy barks!"

myDog.fetch(); // Output: "Buddy fetches the ball."

**112) How does the ‘extends’ keyword work in JavaScript class inheritance?**

The "extends" keyword in JavaScript class inheritance creates a subclass that inherits properties and methods from a parent class, and can also define new properties and methods.

**113) What is a superclass and a subclass?**

A superclass is a class that is inherited from by other classes, while a subclass is a class that inherits properties and methods from a superclass.

**114) What are the advantages and disadvantages of using class inheritance in JavaScript?**

Advantages include code reusability, well-structured, easier maintenance, and encapsulation. Disadvantages include tight coupling, the potential for complexity, inflexibility, unintended side effects, and difficulty managing multiple inheritances (requires mixins).

**115) Can you name the six flags used in regular expressions?**

Regular expressions may have flags that affect the search. There are only 6 of them in JavaScript, including the i, g, m, s, u, and y regular expression flags:

Flag Descriptions:

With the i flag the search is set to case insensitive.

The g flag the search is set to global and looks for all matches, without it only the first search match is returned.

The m flag represents a query in multiline mode.

The s flag enables dot all mode allows a dot to match the newline character.

The u enables full Unicode support, the flag enables correct processing of surrogate pairs.

Sticky flag y sets the stick mode that allows search for the exact position of the text.

**116) What makes JavaScript unique?**

JavaScript is unique because it fully integrates with HTML and CSS, and all major browsers support it. JavaScript is the only browser technology that combines, all three of these features. That distinguishes JavaScript, and explains why it is the most widely used technology for designing browser interfaces.

**117) What is the difference between prototypal and classical inheritance?**

In prototypal inheritance, objects inherit directly from other objects, whereas in classical inheritance, objects inherit from classes. JavaScript uses prototypal inheritance.

**118) What is Explicit Conversion?**

Here, we're explicitly converting a string to a number using the Number() function before adding it to another number.

Example:let str = '10';

let num = 5;

let result = Number(str) + num; // str is explicitly converted to a number.

console.log(result); // Output: 15

**119) Can you explain the difference between a Binary, and Unary operand?**

There are two types of mathematical operators: unary and binary. Unary operators act with a single operand, and Binary operators perform actions with two operands. Unary operators are arithmetic operators that act on a single operand.

Example of Unary Operand:

let x = 1;

x = -x;

console.log(x); // -1, unary negation was applied

Example of Binary Operands:let x = 1,

y = 3;

console.log(y - x); // 2, binary minus subtracts values

**120) What are loops used for in JavaScript?**

Loops in JavaScript are used to execute a block of code repeatedly until a certain condition is met or a specified number of iterations have been completed.

**121) Can you explain how a while loop works?**

A while loop in JavaScript repeatedly executes a block of code as long as a specified condition remains true, and stops when the condition becomes false.

Example: let i = 0;

while (i < 3) {

// shows 0, then 1, then 2

console.log(i);

i++;

}

**122) What is the difference between a Do-While and a While Loop?**

The main difference is that a do-while loop executes its body at least once before checking the condition. A While loop, on the other hand, only executes its loop body if the loop condition is initially true.

Example: do {

// loop body

} while (condition);

**123) How does a for-loop function in JavaScript?**

The for loop defines an initial point, condition, and steps. The initial point runs once when entering the loop. Before each iteration, the condition is evaluated. The loop body repeats while the condition remains true; if false, the loop then exits before the next iteration.

Example: for (let i = 0; i < 3; i++) console.log(i)

**124) What is the difference between the break statement and the continue directive?**

In JavaScript, The break statement terminates the entire loop, while the continue directive skips the current iteration and proceeds to the next iteration in the loop.

**125) What is a switch statement in JavaScript?**

A switch statement in JavaScript is a control flow statement that evaluates an expression and executes a specific block of code based on the matched case.

Example: let x = 'value2';

switch(x) {

case 'value1': // if (x === 'value1')

...

[break]

case 'value2': // if (x === 'value2')

...

[break]

default:

...

[break]

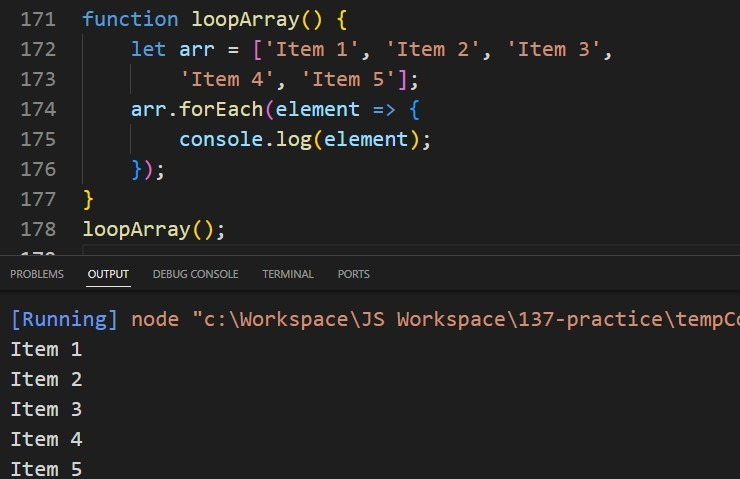
}

**126) What is the purpose of the break statement in a switch statement?**

The `break` statement in a JavaScript `switch` statement is used to prevent the code from running into the next `case` once a match is found and its code has been executed.

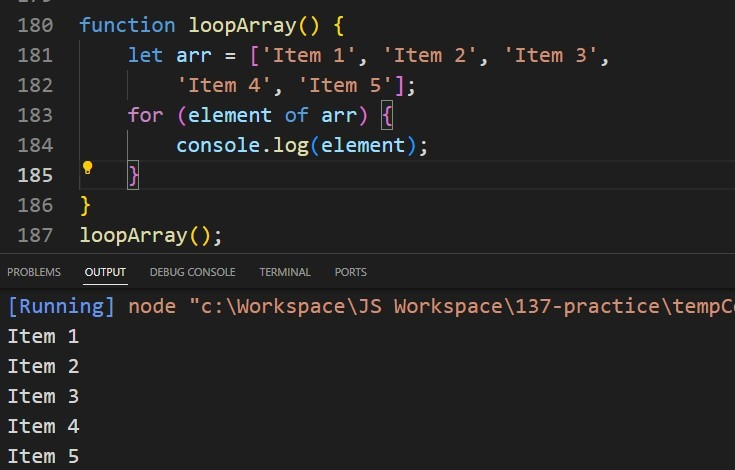
**127) What are the different types of for loops used in js to iterate arrays and objects?**

=> Using the forEach() method: The forEach() method is used to execute code for each element in the array by looping through each of them.

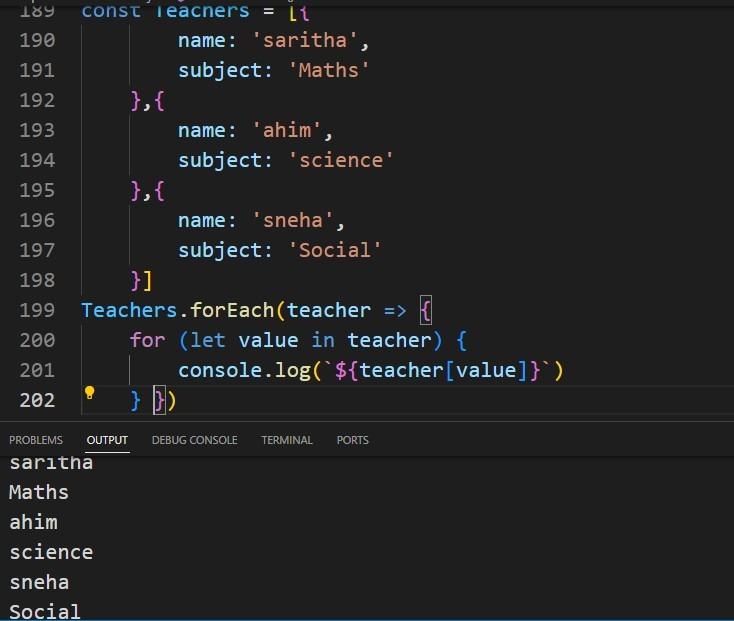
Example:

=> Using the for…of statement: The for…of statement can be used to loop through iterable objects and perform the required functions. Iterable objects include arrays, strings, and other array-like objects.

Example:

=> Using for…in loop: The properties of the object can be iterated over using a for..in loop. The value of each key of the object can be found by using the key as the index of the object.

Example:



Node

**1) What is Node.js? Where can you use it?**

A) Node.js is an open-source, cross-platform JavaScript runtime environment and library to run web applications outside the client’s browser. It is used to create server-side web applications.

**2) Why use Node.js?**

A) Node.js makes building scalable network programs easy. Some of its advantages include:

-It is generally fast.

-It rarely blocks.

-It offers a unified programming language and data type.

-Everything is asynchronous.

**3) What is npm?**

A) npm stands for Node Package Manager, responsible for managing all the packages and modules for Node.js.

Node Package Manager provides two main functionalities:

-Provides online repositories for node.js packages/modules, which are searchable on search.nodejs.org.

-Provides command-line utility to install Node.js packages and also manages Node.js versions and dependencies

**4) What are the modules in Node.js?**

A) Modules are like JavaScript libraries that can be used in a Node.js application to include a set of functions. To include a module in a Node.js application, use the require() function with the parentheses containing the module's name.

**5) Why is Node.js preferred over other backend technologies like Java and PHP?**

A) Some of the reasons why Node.js is preferred include:

-Node.js is very fast.

-Node Package Manager has over 50,000 bundles available at the developer’s disposal

-Perfect for data-intensive, real-time web applications, as Node.js never waits for an API to return data.

-Better synchronization of code between server and client due to same code base

-Easy for web developers to start using Node.js in their projects as it is a JavaScript library

**6) Which database is more popularly used with Node.js?**

A) MongoDB is the most common database used with Node.js. It is a NoSQL, cross-platform, document-oriented database that provides high performance, high availability, and easy scalability.

**7) What are some of the most commonly used libraries in Node.js?**

A) There are two commonly used libraries in Node.js:

-ExpressJS - Express is a flexible Node.js web application framework that provides a wide set of features to develop web and mobile applications.

-Mongoose - Mongoose is also a Node.js web application framework that makes it easy to connect an application to a database.

**8) What is the package.json file?**

A) The package.json file is the heart of a Node.js system. This file holds the metadata for a particular project. The package.json file is found in the root directory of any Node application or module.

This is what a package.json file looks like immediately after creating a Node.js project using the command: npm init

You can edit the parameters when you create a Node.js project.

node-npm.

**9) Explain the concept of middleware in Node.js.**

A) Middleware is a function that receives the request and response objects. Most tasks that the middleware functions perform are:

-Execute any custom code before actual request processing

-Update or modify the request and the response objects

-Finish the request-response cycle

-Invoke the next middleware in the stack if any

**10) What are the different types of HTTP requests?**

A) HTTP defines a set of request methods used to perform desired actions. The request methods include:

-> GET: Used to retrieve the data.

-> POST: Generally used to make a change in state or reactions on the server.

-> DELETE: Used to delete the predetermined resource.

**11) How do you install and delete a dependency ?**

A) For installing dependencies we use npm install or npm i

Example: npm install express

npm install mongoose

npm install nodemon --save-dev // Intalling for user dependency

For deleting we use npm uninstall

Example: npm uninstall express

**12) What are some commonly used timing features of Node.js?**

A) Following is a list of some commonly used timing features of Node.js :

->setTimeout/clearTimeout: This timing feature of Node.js is used to implement delays in the code execution.

->setInterval/clearInterval: The setInterval or clearInterval timing feature is used to run a code block multiple times in the application.

->setImmediate/clearImmediate: This timing feature of Node.js is used to set the execution of the code at the end of the event loop cycle.

->nextTick: This timing feature sets the execution of code at the beginning of the next event loop cycle.

**13) What are buffers in Node.js?**

A) In general, a buffer is a temporary memory mainly used by the stream to hold on to some data until it is consumed. Buffers are used to represent a fixed-size chunk of memory allocated outside of the V8 JavaScript engine. It can't be resized.

**14) Is it possible to access DOM in Node?**

A) No, it is not possible to access DOM in Node.

**15) How to connect NodeJS to Mongoose?**

A) MongoDB is a NoSQL database used to store large amounts of data without any traditional relational database table. Instead of rows & columns, MongoDB used collections & documents to store data. A collections consist of a set of documents & a document consists of key-value pairs which are the basic unit of data in MongoDB.

We should make sure that MongoDB installs on your pc. =>npm install mongoose

To connect a Node.js application to MongoDB, we have to use a library called Mongoose.

=>const mongoose = require("mongoose");

After that, we have to call the connect method of Mongoose

Example: mongoose.connect("mongodb+srv://username:password@cluster0.5qlvwpp.mongodb.net/Blog? retryWrites=true&w=majority")

.then(() => app.listen(5000))

.then(() => console.log("Connected to Database and listening to localhost 5000"))

.catch((err) => console.log(err))

Then we have to define a schema. A schema is a structure, that gives information about how the data is being stored in a collection.

Example: Suppose we want to store information from a contact form of a website.

=>const contactSchema = {

email: String,

query: String,

};

Then we have to create a model using that schema which is then used to store data in a document as objects.

=>const Contact = mongoose.model("Contact", contactSchema);

**16) How to connect NodeJS to MySQL?**

A) First, create a folder for storing the node.js app e.g., node-mysql and use the npm init command to create the package.json file:

=>npm init

Second, install node.js for the MySQL package by using the following command:

=>npm install mysql

Third, create the connect.js inside of the node-mysql folder for storing the code that connects to the MySQL database server.

We should create the database in your MySQL database server by running the following CREATE DATABASE statement:

CREATE DATABASE app;

Code language: SQL (Structured Query Language) (sql)

Once the database is created, you are ready to connect to it from the Node.js application.

Connecting to MySQL database server from node.js

First, import the mysql module by using the following statement:

let mysql = require('mysql');

Code language: JavaScript (javascript)

Second, create a connection to the MySQL database by calling the createConnection() method and providing detailed information on MySQL server such as host, user, password, and database as follows:

let connection = mysql.createConnection({

host: 'localhost',

user: 'root',

password: 'root123',

database: 'app'

});

In this example, we created a connection to app database in the local database server.

Third, call the connect() method on the connection object to connect to the MySQL database server:

connection.connect(function(err) {

if (err) {

return console.error('error: ' + err.message);

}

console.log('Connected to the MySQL server.');

});

The connect() method accepts a callback function that has the err argument which provides the detailed error if any error occurs.

Let’s test connect.js program.

=> node connect.js

Connected to the MySQL server

Code language: JavaScript (javascript)

If you see the message “connected to the MySQL server”, then congratulations, you have been successfully connected to the MySQL database server from the node.js application.

Suppose the todoapps database does not exist in the database server and you try to connect to it, you will get an error message:

=> node connect.js

error: ER\_BAD\_DB\_ERROR: Unknown database 'apps'

Notice that every method which you invoke on the connection object is queued and executed in sequence.

Example: const db = mysql.createConnection({

host: process.env.DATABASE\_HOST,

user: process.env.DATABASE\_USER,

password: process.env.DATABASE\_PASSWORD,

database: process.env.DATABASE

})

**17) How to execute mysql query in nodejs?**

A) The steps for querying data in the MySQL database from a node.js application are as follows:

Establish a connection to the MySQL database server.

\*Execute a SELECT statement and process the result set.

\*Close the database connection.

\*Executing a simple query.

The following select.js program selects all data from the todos table of the app database:

let mysql = require('mysql');

let config = require('./config.js');

let connection = mysql.createConnection(config);

let sql = `SELECT \* FROM todos`;

connection.query(sql, (error, results, fields) => {

if (error) {

return console.error(error.message);

}

console.log(results);

});

**Node**

**1) What is Node.js? Where can you use it?**

A) Node.js is an open-source, cross-platform JavaScript runtime environment and library to run web applications outside the client’s browser. It is used to create server-side web applications.

**2) Why use Node.js?**

A) Node.js makes building scalable network programs easy. Some of its advantages include:

-It is generally fast.

-It rarely blocks.

-It offers a unified programming language and data type.

-Everything is asynchronous.

**3) What is npm?**

A) npm stands for Node Package Manager, responsible for managing all the packages and modules for Node.js.

Node Package Manager provides two main functionalities:

-Provides online repositories for node.js packages/modules, which are searchable on search.nodejs.org.

-Provides command-line utility to install Node.js packages and also manages Node.js versions and dependencies

**4) What are the modules in Node.js?**

A) Modules are like JavaScript libraries that can be used in a Node.js application to include a set of functions. To include a module in a Node.js application, use the require() function with the parentheses containing the module's name.

**5) Why is Node.js preferred over other backend technologies like Java and PHP?**

A) Some of the reasons why Node.js is preferred include:

-Node.js is very fast.

-Node Package Manager has over 50,000 bundles available at the developer’s disposal

-Perfect for data-intensive, real-time web applications, as Node.js never waits for an API to return data.

-Better synchronization of code between server and client due to same code base

-Easy for web developers to start using Node.js in their projects as it is a JavaScript library

**6) Which database is more popularly used with Node.js?**

A) MongoDB is the most common database used with Node.js. It is a NoSQL, cross-platform, document-oriented database that provides high performance, high availability, and easy scalability.

**7) What are some of the most commonly used libraries in Node.js?**

A) There are two commonly used libraries in Node.js:

-ExpressJS - Express is a flexible Node.js web application framework that provides a wide set of features to develop web and mobile applications.

-Mongoose - Mongoose is also a Node.js web application framework that makes it easy to connect an application to a database.

**8) What is the package.json file?**

A) The package.json file is the heart of a Node.js system. This file holds the metadata for a particular project. The package.json file is found in the root directory of any Node application or module.

This is what a package.json file looks like immediately after creating a Node.js project using the command: npm init

You can edit the parameters when you create a Node.js project.

node-npm.

**9) Explain the concept of middleware in Node.js.**

A) Middleware is a function that receives the request and response objects. Most tasks that the middleware functions perform are:

-Execute any code

-Update or modify the request and the response objects

-Finish the request-response cycle

-Invoke the next middleware in the stack

**10) What are the different types of HTTP requests?**

A) HTTP defines a set of request methods used to perform desired actions. The request methods include:

-> GET: Used to retrieve the data.

-> POST: Generally used to make a change in state or reactions on the server.

-> DELETE: Used to delete the predetermined resource.

**11) How do you install and delete a dependency ?**

A) For installing dependencies we use npm install or npm i

Example: npm install express

npm install mongoose

npm install nodemon --save-dev // Intalling for user dependency

For deleting we use npm uninstall

Example: npm uninstall express

**12) What are some commonly used timing features of Node.js?**

A) Following is a list of some commonly used timing features of Node.js :

->setTimeout/clearTimeout: This timing feature of Node.js is used to implement delays in the code execution.

->setInterval/clearInterval: The setInterval or clearInterval timing feature is used to run a code block multiple times in the application.

->setImmediate/clearImmediate: This timing feature of Node.js is used to set the execution of the code at the end of the event loop cycle.

->nextTick: This timing feature sets the execution of code at the beginning of the next event loop cycle.

**13) What are buffers in Node.js?**

A) In general, a buffer is a temporary memory mainly used by the stream to hold on to some data until it is consumed. Buffers are used to represent a fixed-size chunk of memory allocated outside of the V8 JavaScript engine. It can't be resized.

**14) Is it possible to access DOM in Node?**

A) No, it is not possible to access DOM in Node.

**15) How to connect NodeJS to Mongoose?**

A) MongoDB is a NoSQL database used to store large amounts of data without any traditional relational database table. Instead of rows & columns, MongoDB used collections & documents to store data. A collections consist of a set of documents & a document consists of key-value pairs which are the basic unit of data in MongoDB.

We should make sure that MongoDB installs on your pc. =>npm install mongoose

To connect a Node.js application to MongoDB, we have to use a library called Mongoose.

=>const mongoose = require("mongoose");

After that, we have to call the connect method of Mongoose

Example: mongoose.connect("mongodb+srv://username:password@cluster0.5qlvwpp.mongodb.net/Blog? retryWrites=true&w=majority")

.then(() => app.listen(5000))

.then(() => console.log("Connected to Database and listening to localhost 5000"))

.catch((err) => console.log(err))

Then we have to define a schema. A schema is a structure, that gives information about how the data is being stored in a collection.

Example: Suppose we want to store information from a contact form of a website.

=>const contactSchema = {

email: String,

query: String,

};

Then we have to create a model using that schema which is then used to store data in a document as objects.

=>const Contact = mongoose.model("Contact", contactSchema);

**16) How to connect NodeJS to MySQL?**

A) First, create a folder for storing the node.js app e.g., node-mysql and use the npm init command to create the package.json file:

=>npm init

Second, install node.js for the MySQL package by using the following command:

=>npm install mysql

Third, create the connect.js inside of the node-mysql folder for storing the code that connects to the MySQL database server.

We should create the database in your MySQL database server by running the following CREATE DATABASE statement:

CREATE DATABASE app;

Code language: SQL (Structured Query Language) (sql)

Once the database is created, you are ready to connect to it from the Node.js application.

Connecting to MySQL database server from node.js

First, import the mysql module by using the following statement:

let mysql = require('mysql');

Code language: JavaScript (javascript)

Second, create a connection to the MySQL database by calling the createConnection() method and providing detailed information on MySQL server such as host, user, password, and database as follows:

let connection = mysql.createConnection({

host: 'localhost',

user: 'root',

password: 'root123',

database: 'app'

});

In this example, we created a connection to app database in the local database server.

Third, call the connect() method on the connection object to connect to the MySQL database server:

connection.connect(function(err) {

if (err) {

return console.error('error: ' + err.message);

}

console.log('Connected to the MySQL server.');

});

The connect() method accepts a callback function that has the err argument which provides the detailed error if any error occurs.

Let’s test connect.js program.

=> node connect.js

Connected to the MySQL server

Code language: JavaScript (javascript)

If you see the message “connected to the MySQL server”, then congratulations, you have been successfully connected to the MySQL database server from the node.js application.

Suppose the todoapps database does not exist in the database server and you try to connect to it, you will get an error message:

=> node connect.js

error: ER\_BAD\_DB\_ERROR: Unknown database 'apps'

Notice that every method which you invoke on the connection object is queued and executed in sequence.

Example: const db = mysql.createConnection({

host: process.env.DATABASE\_HOST,

user: process.env.DATABASE\_USER,

password: process.env.DATABASE\_PASSWORD,

database: process.env.DATABASE

})

**17) How to execute mysql query in nodejs?**

A) The steps for querying data in the MySQL database from a node.js application are as follows:

Establish a connection to the MySQL database server.

\*Execute a SELECT statement and process the result set.

\*Close the database connection.

\*Executing a simple query.

The following select.js program selects all data from the todos table of the app database:

let mysql = require('mysql');

let config = require('./config.js');

let connection = mysql.createConnection(config);

let sql = `SELECT \* FROM todos`;

connection.query(sql, (error, results, fields) => {

if (error) {

return console.error(error.message);

}

console.log(results);

});

**Mongoose**

**1) What is Mongoose?**

A) Mongoose is a MongoDB object modeling tool designed to work in an asynchronous environment. Mongoose supports both promises and callbacks.

**2) Can you explain how to connect to a MongoDB using Mongoose?**

A) In order to connect to a MongoDB using Mongoose, you will need to first install the Mongoose npm package. Once you have done that, you can use the Mongoose.connect() function to connect to your MongoDB database.

**3) Can you give me an example of how you would define a schema in Mongoose?**

A) A schema in Mongoose is simply a representation of the structure of your data. For example, if you were creating a schema for a blog post, it might look something like this:

“`

var blogSchema = new mongoose.Schema({

title: String,

body: String,

date: { type: Date, default: Date.now },

comments: [{ type: mongoose.Schema.Types.ObjectId, ref: ‘Comment’ }]

});

“`

**4) How do the Schema and Model objects work together?**

A) The Schema object contains information about the structure of the data in the collection, while the Model object contains functions that allow you to interact with the data in the collection.

**5) How can we use mongoose models to create, read, update, and delete documents from our database?**

A) We can use mongoose models to create, read, update, and delete documents from our database by using the model functions create(), find(), update(), and deleteOne().

**6) How do we specify validation rules for data fields when defining our schema?**

A) We can specify validation rules for data fields by adding validation keywords to the field definition in our schema. For example, we could add the “required” keyword to a field to make sure that it is always populated with a value.

**7) In what situations is it best to use a Mongoose virtual field over a normal one?**

A) A Mongoose virtual field is best used when you want to define a field that is not actually stored in the MongoDB database, but that you want to be able to access and manipulate as if it were a normal field. This can be useful for computed fields, or for fields that you want to be able to populate from an outside source.

**8) What’s the difference between static methods and instance methods in Mongoose?**

A) Static methods are methods that are called on the model itself, while instance methods are methods that are called on documents that are retrieved from the database. For example, you might have a static method on the User model called findByEmail that takes an email address and returns the user with that email address. An instance method might be one that returns the user’s full name.

**9) What are some ways to validate Mongoose schemas?**

A) There are several ways to validate Mongoose schemas. One way is to use the built-in validators that Mongoose provides, such as the “required” validator. Another way is to use custom validation functions. Finally, you can also use a third-party validation library like Validator.js.

**10) What are middleware functions in Mongoose?**

A) Middleware functions in Mongoose are functions that are run before or after certain operations are executed. For example, you could use a middleware function to run some code before a document is saved to the database. This would allow you to do things like validate the data or perform some other operation on it before it is actually stored.

**15) What’s the difference between findOne() and findById()?**

A) The findOne() function will return the first document that matches the query. The findById() function will return the document with the specified id.

**16) What is the purpose of $where in Mongoose queries?**

A) The $where operator in Mongoose queries allows you to execute arbitrary JavaScript expressions to query for documents. This can be useful if you need to query for documents based on complex criteria that can’t be easily expressed using the other Mongoose query operators. However, you should be aware that using $where can be very slow, since it has to execute the JavaScript expression for every document in the collection.

**EXPRESS**

**1) What is Express.js?**

A) Express.js, or simply Express, is a free, open-source, lightweight, and fast backend web application framework for Node.js. It is released as open-source software under the MIT License.

**2) Is Express.js front-end or backend framework?**

A) Express.js or Express is a JavaScript backend framework. It is mainly designed to develop complete web applications (single-page, multi-page, and hybrid web applications) and APIs. Express is the backend component of the MEAN stack where M stands for MongoDB, which handles database; E stands for Express, which handles backend; A stands for AngularJS, which is for the front-end, and N stands for Node.

**3) Why do we use Express.js?**

A) A Express.js is an automatically prebuilt Node.js framework that facilitates us to create server-side web applications faster and smarter. The main reason for choosing Express is its simplicity, minimalism, flexibility, and scalability characteristics.

**4) How does an Express code look like ?**

A) The express.js program is saved with ".js" extension.

See the example:var express = require('express');

var app = express();

app.get('/', function (req, res) {

res.send('Welcome to Express!');

});

var server = app.listen(8000, function () {

var host = server.address().address;

var port = server.address().port;

console.log('Example app listening at http://%s:%s', host, port);

});

When you run the Node.js command prompt, the app will listen at the specified server address and give the following output.

Output: Welcome to Express

**5) Write a code to get post a query in Express.js.**

A) var bodyParser = require('body-parser')

app.use( bodyParser.json() ); // to support JSON-encoded

app.use(bodyParser.urlencoded({ // to support URL-encoded

extended: true

}));

**6) How can you deal with error handling in Express.js? Explain with an examples.**

A) Error handling is much easier in the Express versions over Express 4.0. Use the following steps to do the error handling:

Create an Express.js application. There is no built-in middleware like error handler in express 4.0, so you have to either install a middleware or create a custom one.

Create a middleware as following:

// error handler

app.use(function(err, req, res, next) {

// set locals, only providing error in development

res.locals.message = err.message;

res.locals.error = req.app.get('env') === 'development' ? err : {};

// render the error page

res.status(err.status || 500);

res.render('error');

});

Install Error Handler Middleware as following:

npm install errorhandler --save

Create a variable:

var errorhandler = require('errorhandler')

Use the middleware as following:

if (process.env.NODE\_ENV === 'development') {

// only use in development

app.use(errorhandler({log: errorNotification}))

}

function errorNotification(err, str, req) {

var title = 'Error in ' + req.method + ' ' + req.url

notifier.notify({

title: title,

message: str

})

}

**7) Write the code to start serving static files in Express.js?**

A) app.use(express.static('public'))

app.use('/static', express.static(path.join(\_\_dirname, 'public')))

**8) What is Middleware in Express.js?**

A) Middleware is a function invoked by the Express routing layer before the final request handler.

Middleware functions are used to perform the following tasks:

-> It is used to execute any code.

-> It is also used to make changes to the request and the response objects.

-> It is responsible for ending the request-response cycle.

-> It can call the next middleware function in the stack.

**9) What are the different types of Middleware?**

A) Type of Middleware are:

-> Application-level Middleware

-> Router-level Middleware

-> Error-handling Middleware

-> Built-in Middleware

-> Third-party Middleware

=> Application-level middleware: The application-level middleware method is used to bind to the app object using app.use() method. It applies on all routes.

//This middleware will execute for each route.

app.use(function (req, res, next) {

console.log('Current Time:', Date.now())

next()

})

=> Router-level Middleware: The router-level Middleware is used to bind to a specific instance of express.Router().Built-in Middleware: The built-in Middleware was introduced with version 4.x. It ends the dependency on Connect.

There are the following built-in middleware functions in Express.js:

-> static: It is used to serve static assets such as HTML files, images, etc.

-> json: It is used to parse the incoming requests with JSON payloads. It is available with Express 4.16.0+

-> urlencoded: It is used to parse the incoming requests with URL-encoded payloads. It is available with Express 4.16.0+

=> Third-party Middleware: There are many third-party middleware available such as:

-> Body-parser

-> Cookie-parser

-> Mongoose

-> Sequelize

-> Cors

-> Express-validator

Example: var bodyParser = require('body-parser');

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: false }))