INTERVIEW QUESTIONS

1. What are pipes?

A pipe takes in data as input and transforms it to a desired output. You can chain pipes together in potentially useful combinations. You can write your own custom pipes. Angular comes with a stock of pipes such as DatePipe, Upper CasePipe, LowerCasePipe, CurrencyPipe, and PercentPipe.

2. What is the minimum definition of a component?

The absolute minimal configuration for a @Component in Angular is a template. Both template properties are set to optional because you have to define either template or templateUrl.

3. What's the difference between an Angular component and module?

Components control views (html). They also communicate with other components and services to bring functionality to your app.

Modules consist of one or more components. They do not control any html. Your modules declare which components can be used by components belonging to other modules, which classes will be injected by the dependency injector and which component gets bootstrapped. Modules allow you to manage your components to bring modularity to your app.

4. How can I select an element in a component template?

You can get a handle to the DOM element via ElementRef by injecting it into your component's constructor

5. What is an observer?

Observer is an interface for a consumer of push-based notifications delivered by an Observable. It has below structure,

```
interface Observer<T> {
    closed?: boolean;
    next: (value: T) => void;
    error: (err: any) => void;
    complete: () => void;
```

```
  }
```

A handler that implements the Observer interface for receiving observable notifications will be passed as a parameter for observable as below,

```
myObservable.subscribe(myObserver);
```

6. What is an observable?

An Observable is a unique Object similar to a Promise that can help manage async code. Observables are not part of the JavaScript language so we need to rely on a popular Observable library called RxJS. The observables are created using new keyword. Let see the simple example of observable,

7. What is Redux and how does it relate to an Angular app?

Redux is a way to manage application state and improve maintainability of asynchronicity in your application by providing a single source of truth for the application state, and a unidirectional flow of data change in the application. ngrx/store is one implementation of Redux principles.

8. What are the Core Dependencies of Angular 7?

RxJS 6.3, TypeScript 3.1

9. Why Incremental DOM Has Low Memory Footprint?

Virtual DOM creates a whole tree from scratch every time you rerender.

Incremental DOM, on the other hand, doesn't need any memory to rerender the view if it doesn't change the DOM. We only have to allocate the memory when the DOM nodes are added or removed. And the size of the allocation is proportional to the size of the DOM change.

10. What are the ways to control AOT compilation?

1. By providing template compiler options in the tsconfig.json file
2. By configuring Angular metadata with decorators

11. What is activated route?

ActivatedRoute contains the information about a route associated with a component loaded in an outlet. It can also be used to traverse the router state tree. The ActivatedRoute will be injected as a router service to access the information.

12. What is router outlet?

The RouterOutlet is a directive from the router library and it acts as a placeholder that marks the spot in the template where the router should display the components for that outlet. Router outlet is used as a component,

13. What are the utility functions provided by RxJS?

The RxJS library also provides below utility functions for creating and working with observables.

1. Converting existing code for async operations into observables
2. Iterating through the values in a stream
3. Mapping values to different types
4. Filtering streams
5. Composing multiple streams

14. What is multicasting?

Multi-casting is the practice of broadcasting to a list of multiple subscribers in a single execution. Let's demonstrate the multi-casting feature

15. What is subscribing?

An Observable instance begins publishing values only when someone subscribes to it. So you need to subscribe by calling the subscribe() method of the instance, passing an observer object to receiving the notifications.

16. What is the difference between *ngIf vs [hidden]?

*ngIf effectively removes its content from the DOM while [hidden] modifies the display property and only instructs the browser to not show the content but the DOM still contains it.

17. What is the difference between "@Component" and "@Directive" in Angular?

Directives add behaviour to an existing DOM element or an existing component instance.

- A component, rather than adding/modifying behaviour, actually creates its own view (hierarchy of DOM elements) with attached behaviour.

Write a component when you want to create a reusable set of DOM elements of UI with custom behaviour. Write a directive when you want to write reusable behaviour to supplement existing DOM elements.

18. How would you protect a component being activated through the router?

The Angular router ships with a feature called guards. These provide us with ways to control the flow of our application. We can stop a user from visiting certain routes, stop a user from leaving routes, and more. The overall process for protecting Angular routes:

- Create a guard service: ng g guard auth
- Create canActivate() or canActivateChild() methods

19. What is the difference between Structural and Attribute directives in Angular?

- Structural directives are used to alter the DOM layout by removing and adding DOM elements. It is far better in changing the structure of the view. Examples of Structural directives are NgFor and Nglf.

- Attribute Directives These are being used as characteristics of elements. For example, a directive such as built-in NgStyle in the template Syntax guide is an attribute directive.

20. What is the purpose of base href tag?

The routing application should add element to the index.html as the first child in the tag inorder to indicate how to compose navigation URLs. If app folder is the application root then you can set the href value as below

21. What is a bootstrapping module?

Every application has at least one Angular module, the root module that you bootstrap to launch the application is called as bootstrapping module. It is commonly known as AppModule.

22. What is interpolation?

Interpolation is a special syntax that Angular converts into property binding. It's a convenient alternative to property binding. It is represented by double curly braces({{}}). The text between the braces is often the name of a component property. Angular replaces that name with the string value of the corresponding component property.

23. Explain the difference between `Promise` and `Observable` in Angular?

Promise emits a single value while Observable emits multiple values. So, while handling a HTTP request, Promise can manage a single response for the same request, but what if there are multiple responses to the same request, then we have to use Observable.

24. Explain the difference between "Constructor" and "ngOnInit"?

The Constructor is a default method of the class that is executed when the class is instantiated and ensures proper initialisation of fields in the class and its subclasses. Angular, or better Dependency Injector (DI), analyzes the constructor parameters and when it creates a new instance by calling new MyClass() it tries to find providers that match the types of the constructor parameters, resolves them and passes them to the constructor

ngOnInit is a life cycle hook called by Angular to indicate that Angular is done creating the component.

We have to import OnInit like this in order to use it (actually implementing OnInit is not mandatory but considered good practice):

25. What is difference between "declarations", "providers" and "import" in NgModule?

imports makes the exported declarations of other modules available in the current module

declarations are to make directives (including components and pipes) from the current module available to other directives in the current module. Selectors of directives, components or pipes are only matched against the HTML if they are declared or imported.

providers are to make services and values known to DI (dependency injection). They are added to the root scope and they are injected to other services or directives that have them as dependency.

26. Features of Angular 7 are:

CLI Prompts, Application Performance, Virtual Scrolling, Drag and Drop, Angular Compiler, Angular Do-Bootstrap, Better Error Handling

27. What is a template in Angular7?

In Angular 7, templates are written with the help of HTML that contains Angular JS specification elements and attributes. The main role of Angular 7 is to combine the templates which ultimately control the dynamic view that is visible to the users in the browsers.

28. What is AOT?

The Angular Ahead-of-Time (AOT) compiler converts your Angular HTML and TypeScript code into efficient JavaScript code during the build phase before the browser downloads and runs that code. ... The browser loads executable code so it can render the application immediately, without waiting to compile the app first.

29. What is Angular Universal?

Angular Universal is the process of server-side rendering (SSR) your application to HTML on the Server (ie: Node. js). Typical Angular applications are Single-Page Applications (aka SPA's) where the rendering occurs on the Browser. This process can also be referred to as client-side rendering (CSR).

30. What is a parameterized pipe?

A pipe can accept any number of optional parameters to fine-tune its output. To add parameters to a pipe, follow the pipe name with a colon (:) and then the parameter value (such as currency:'EUR'). If the pipe accepts multiple parameters, separate the values with colons (such as slice:1:5)

31. What is Zone in Angular?

In simple language Zone.js is a api or set of programs which is used by angular 2 to update the application view when any change occurred. A Zone is an execution context that persists across asynchronous task. for example:Events, XMLHttpRequests and Timers(setTimeout(), setInterval()) etc.

32. What does a just-in-time (JIT) compiler do (in general)?

A Just-In-Time (JIT) compiler is a feature of the run-time interpreter, that instead of interpreting bytecode every time a method is invoked, will compile the bytecode into the machine code instructions of the running machine, and then invoke this object code instead.

33. Why would you use lazy loading modules in Angular app?

Lazy Loading is the technique of loading the module or data on demand. It helps us to better the application performance and reduce the initial bundle size of our files. The initial page loads faster and we can also split the application into the logic chunks which can be loaded on demand.

34. What are the advantages with AOT?

1. Faster download: - The Angular 2 app is already compiled so it is faster. 2. Faster Rendering: - If the app is not AOT compiled and the compilation process happens in the browser once the application is fully loaded.

35. What is the difference between pure and impure pipe?

A pure pipe is only called when Angular detects a change in the value or the parameters passed to a pipe. An impure pipe is called for every change detection cycle no matter whether the value or parameter(s) changes.

36. What is the difference between BehaviorSubject vs Observable?

Observable is a Generic, and BehaviorSubject is technically a sub-type of Observable because BehaviorSubject is an observable with specific qualities.

37. Why did the Google team go with incremental DOM instead of virtual DOM?

They have one goal in mind: applications have to perform well on mobile devices. This mainly meant optimizing two things: the bundle size and the memory footprint. ... The rendering engine has to have low memory footprint.

38. Explain the Architecture overview of Angular.

- Component
- Templates
- Metadata
- Data Binding
- Directives
- Services
- Dependency Injection

39. How would you update Angular 6 to Angular 7?

ng update @angular/cli @angular/core

40. What is the UrlSegment Interface in Angular 7?

The UrlSegment is a part of a URL between the two slashes and it contains a path and matrix parameters associated with the segment.

41. What is Do Bootstrap (ng Do Bootstrap) In Angular 7?

The ng Do Bootstrap is a new life-cycle hook added in Angular 7 and Do Bootstrap is an interface.

42. What is IVY Renderer? Is it supported by Angular 7?

Angular will be releasing a new kind of rendering pipeline and view engine. The purpose of angular view engine is to translate the templates and components that we have written into the regular HTML and JavaScript so it is easy for the browser to read it comfortably. Ivy is believed to be splendid for the Angular Renderer.

Yes, it is supported by Angular 7.

43. What is TestBed in Angular?

Angular Test Bed (ATB) is the primary API for writing unit tests for Angular Application and libraries. It allows us to test behaviors and change detections that depend on the Angular Framework

44. What Is Node.js?

Node.js is a powerful framework developed on Chrome's V8 JavaScript engine that compiles the JavaScript directly into the native machine code. It is a lightweight framework used for creating server-side web applications and extends JavaScript API to offer usual server-side functionalities. It is generally used for large-scale application development, especially for video streaming sites, single-page application, and other web applications.

45. List down the major benefits of using Node.js?

Fast, Asynchronous, Scalable, Open Source, No Buffering

46. What is an error-first callback in Node.js?

Error-first callbacks in Node.js are used to pass errors and data. The very first parameter you need to pass to these functions has to be an error object while the other parameters represent the associated data. Thus you can pass the error object for checking if anything is wrong and handle it. In case there is no issue, you can just go ahead and with the subsequent arguments.

47. Explain the purpose of module.exports?

A module in Node.js is used to encapsulate all the related codes into a single unit of code which can be interpreted by shifting all related functions into a single file. For example, suppose you have a file called greet.js that contains the two functions as shown below:

48. Explain the purpose of ExpressJS package?

Express.js is a framework built on top of Node.js that facilitates the management of the flow of data between server and routes in the server-side applications. It is a lightweight and flexible framework that provides a wide range of features required for the web as well as mobile application development. Express.js is developed on the middleware module of Node.js called connect. The connect module further makes use of http module to communicate with Node.js. Thus, if you are working with any of the connect based middleware modules, then you can easily integrate with Express.js.

49. What are Streams? List types of streams available in Node Js ?

- Readable − For reading operation.
- Writable − For writing operation.
- Duplex − Used for both read and write operation.
- Transform − A type of duplex stream where the output is computed based on the input.

50. Life Cycle hooks of Angular ?

1. ngOnChanges()
2. ngOnInit()
3. ngDoCheck()
4. ngAfterContentInit()
5. ngAfterContentChecked()
6. ngAfterViewInit()
7. ngAfterViewChecked()
8. ngOnDestroy

51. What is Javascript Closure? Purpose?

A **closure** is the combination of a function bundled together (enclosed) with references to its surrounding state (the **lexical environment**). In other words, a closure gives you access to an outer function's scope from an inner function. In JavaScript, closures are created every time a function is created, at function creation time.

```
function init() {
  var name = 'Mozilla'; // name is a local variable created by init
  function displayName() { // displayName() is the inner function, a closure
    alert(name); // use variable declared in the parent function
  }
  displayName();
}
init();
```

52. Difference between Subject & Behaviour Subject?

BehaviourSubject will return the initial value or the current value on Subscription

Subject does not return the current value on Subscription. It triggers only on .next(value) call and return/output the value

	Each next subscribers receive...
`Subject`	...only upcoming values
`BehaviorSubject`	...one previous value and upcoming values
`ReplaySubject`	...all previous values and upcoming values
`AsyncSubject`	...latest value when stream will close

53. CSS Display Property?

Compared to display: inline, the major difference is that display: inline-block allows to set a width and height on the element.

Also, with **display: inline-block**, the top and bottom margins/paddings are respected, but with display: inline they are not.

Compared to display: block, the major difference is that display: inline-block does not add a line-break after the element, so the element can sit next to other elements.

54. Dependencies vs devDependencies?

Dependencies should contain libs and framewors your app is built on, such as Vue, React, Angular, Express, JQuery and etc. You will agree with me, if I say, that your project wont work without these packages

devDependencies should contain packages which are used during development or which are used to build your bundle, for example, mocha, jscs, grunt-contrib-watch, gulp-jade and etc. These packages are neseccery only while you are developing your project, also ESlint is used to check everything during building your bundle.

55. What is XSS?

Cross-site Scripting (XSS) is a client-side code injection attack. The attacker aims to execute malicious scripts in a web browser of the victim by including malicious code in a legitimate web page or web application.

56. How can you secure your HTTP cookies against XSS attacks?

XSS occurs when the attacker injects executable JavaScript code into the HTML response.
To mitigate these attacks, you have to set flags on the set-cookie HTTP header:

HttpOnly - this attribute is used to help prevent attacks such as cross-site scripting since it does not allow the cookie to be accessed via JavaScript.
Secure - this attribute tells the browser to only send the cookie if the request is being sent over HTTPS.
So it would look something like this: Set-Cookie: sid=<cookie-value>; HttpOnly. If you are using Express, with express-cookie session, it is working by default

57. In JS we follow error-first callback, then why in promise, first func is resolve? why not reject?

Some callbacks can consume more parameters than just two, hence, trying to figure out if err is second, third etc would be confusing, therefore, having err passed as a first one makes sens.

58. What are the two types of API functions in Node.js ?

The two types of API functions in Node.js are

a) Asynchronous, non-blocking functions
b) Synchronous, blocking functions

59. Can you access DOM in node?

No, you cannot access DOM in node.

60. What are the two arguments that async.queue takes?

The two arguments that async.queue takes

a) Task function

b) Concurrency value

61. How Node.js overcomes the problem of blocking of I/O operations?

Node.js solves this problem by putting the event based model at its core, using an event loop instead of threads.

62. What is an event loop in Node.js ?

To process and handle external events and to convert them into callback invocations an event loop is used. So, at I/O calls, node.js can switch from one request to another .

63. Event loop stages ?

timers: this phase executes callbacks scheduled by setTimeout() and setInterval().
pending callbacks: executes I/O callbacks deferred to the next loop iteration.
idle, prepare: only used internally.
poll: retrieve new I/O events; execute I/O related callbacks (almost all with the exception of close callbacks, the ones scheduled by timers, and setImmediate()); node will block here when appropriate.
check: setImmediate() callbacks are invoked here.
close callbacks: some close callbacks, e.g. socket.on('close', ...).

64. What is ‘Callback’ in node.js?

Callback function is used in node.js to deal with multiple requests made to the server. Like if you have a large file which is going to take a long time for a server to read and if you don’t want a server to get engage in reading that large file while dealing with other requests, call back function is used. Call back function allows the server to deal with pending request first and call a function when it is finished.

65. forRoot() vs forChild()

forRoot creates a module that contains all the directives, the given routes, and the router service itself.

forChild creates a module that contains all the directives and the given routes, but does not include the router service.

66. Child process in node js

ished.
https://www.freecodecamp.org/news/node-js-child-processes-everything-you-need-to-know-e69498fe970a/

67. What is a Data Structure?

A data structure is a way of organizing the data so that the data can be used efficiently. Different kinds of data structures are suited to different kinds of applications, and some are highly specialized to specific tasks. For example, B-trees

are particularly well-suited for implementation of databases, while compiler implementations usually use hash tables to look up identifiers.

68. What are linear and non linear data Structures?

Linear: A data structure is said to be linear if its elements form a sequence or a linear list. Examples: Array. Linked List, Stacks and Queues

Non-Linear: A data structure is said to be non-linear if traversal of nodes is nonlinear in nature. Example: Graph and Trees.

69. What are the various operations that can be performed on different Data Structures?

- **Insertion**: Add a new data item in the given collection of data items.
- **Deletion**: Delete an existing data item from the given collection of data items.
- **Traversal**: Access each data item exactly once so that it can be processed.
- **Searching**: Find out the location of the data item if it exists in the given collection of data items.
- **Sorting**: Arranging the data items in some order i.e. in ascending or descending order in case of numerical data and in dictionary order in case of alphanumeric data.

70. How is an Array different from Linked List?

- The size of the arrays is fixed, Linked Lists are Dynamic in size.
- Inserting and deleting a new element in an array of elements is expensive, Whereas both insertion and deletion can easily be done in Linked Lists.
- Random access is not allowed in Linked Listed.
- Extra memory space for a pointer is required with each element of the Linked list.
- Arrays have better cache locality that can make a pretty big difference in performance.

71. What is Stack and where it can be used?

Stack is a linear data structure which the order LIFO(Last In First Out) or FILO(First In Last Out) for accessing elements. Basic operations of stack are : **Push, Pop , Peek**

Applications of Stack:

1. Infix to Postfix Conversion using Stack
2. Evaluation of Postfix Expression
3. Reverse a String using Stack
4. Implement two stacks in an array
5. Check for balanced parentheses in an expression

72. **What is a Queue, how it is different from stack and how is it implemented?**

Queue is a linear structure which follows the order is **F**irst **I**n **F**irst **O**ut (FIFO) to access elements. Mainly the following are basic operations on queue: **Enqueue,**

Dequeue, **Front, Rear**

The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added. Both Queues and Stacks can be implemented using Arrays and Linked Lists.

73. How to write sorting without sort keyword?

```
function sorter(array){
        var swap;
          for (var i = 1; i < array.length; i++){
            if(array[i - 1]>array[i]){
              swap = array[i - 1];
              array[i - 1] = array[i];
              array[i] = swap;
            }
          }
        return array; .filter(v => v)
}
const j = sorter([1,5,2,4,6]);
console.log(j);
```

```
function reverseArr(input) {
```

```
	var ret = new Array;
	for(var i = input.length-1; i >= 0; i--) {
		ret.push(input[i]);
	}
	return ret;
}
var a = [3,5,7,8]
var b = reverseArr(a);
```

74. let a = ‘drWets’; how to convert uppercase into lowercase, lowercase into uppercase without in built function?

Just get the ascii code for each char, and if it ranges between 97 to 122 (a to z) deduct 32 from val to get the corresponding upper case char. Hope this helps.

```
function upperCase(a)

{

    var output="";

    for(var x=0;x<a.length;x++) {

      output+=String.fromCharCode(a.charCodeAt(x)>96 && a.charCodeAt(x)<123 ?
a.charCodeAt(x)-32: a.charCodeAt(x))

    }



    return output;

}
```

```
console.log(upperCase('asdf'));
```

75. Write pollyfil for reduce function. Polyfill means es5 didn't support es6 functions. So write protorype for es5

```
Array.prototype.myReduce = function(fn, initial) {
    let values = this;
    values.forEach(item => {
        initial = initial !== undefined ? fn(initial, item) : item
    });
    return initial;
}
var values = [2, 5, 5]
values.myReduce((a, b) => a * b)  // 50
```

76. what is prototype in javascript?

The prototype is an object that is associated with every functions and objects by default in JavaScript, where function's prototype property is accessible and modifiable and object's prototype property (aka attribute) is not visible.

78. Javascript Pass by value & Pass by reference?

Pass by Value:

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.

Javascript always pass by value so changing the value of the variable never changes the underlying primitive (String or number).

Pass by Reference:

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.

79. How to achieve observable things without rxjs?

Using getter and setter

80. Let vs var ?

The main difference between let and var is that scope of a variable defined with let is limited to the block in which it is declared while variable declared with var has the

global scope. So we can say that var is rather a keyword which defines a variable globally regardless of block scope.

```
let a = 'hello'; // globally scoped
var b = 'world'; // globally scoped
console.log(window.a); // undefined
console.log(window.b); // 'world'
var a = 'hello';
var a = 'world'; // No problem, 'hello' is replaced.
let b = 'hello';
let b = 'world'; // SyntaxError: Identifier 'b' has already been declared
```

81. What is Hoisting?

    In JavaScript, a variable can be declared after it has been used.

    In other words; a variable can be used before it has been declared.

82. Monolithic vs Microservices?

83. How we are doing typescript code?

84. Return type of await?

85. Micro service architecture with same db

86. What is inheritance?

    Inheritance is an important concept in object oriented programming. In the

classical inheritance, methods from base class get copied into derived class.
In JavaScript, inheritance is supported by using prototype object.

87. What is the difference between classical inheritance and prototypal inheritance?

Class Inheritance: Class inheritance is a way for one class to extend another class. So we can create new functionality on top of the existing.

Prototypal Inheritance: instances inherit directly from other objects. Instances are typically instantiated via factory functions or `Object.create()`. Instances may be composed from many different objects, allowing for easy selective inheritance.

88. What is data mutation in JavaScript?

Mutation means a change in the form or nature of the original data. In JavaScript, there are two data types: 1) primitive and 2) non-primitive/reference data types
Primitive types in JavaScripts are immutable, meaning that if we change one then a new instance is created as a result.

Reference data types are Objects and Arrays. Reference types in JavaScript are mutable, meaning that the state and fields of mutable types can be changed. No new instance is created as a result.

89. JavaScript: Primitives/References ?

In JS, there are primitive data types and reference data types. primitive data types

are referenced by value while the non-primitive/reference data types point to memory addresses.

Primitive data types are:

- Boolean
- Number
- String
- Null
- Undefined
- Symbol

Reference data types are:

- Objects
- Arrays

90. What is the CSS Box model and what are its elements?

The CSS box model is used to define the design and layout of elements of CSS.

The elements are:

- Margin - It removes the area around the border. It is transparent.
- Border - It represents the area around the padding
- Padding - It removes the area around the content. It is transparent.
- Content - It represents the content like text, images, etc.

91. What are Pseudo-classes?

A pseudo-class is used to define a special state of an element.

Eg:

```
selector:pseudo-class {
```

```
        property: value;
    }
```

92. What is the difference between logical tags and physical tags?

Physical tags are referred to as presentational markup while logical tags are useless for appearances.

Physical tags are newer versions, on the other hand, logical tags are old and concentrate on content.

93. What is tweening?

It is the process of generating intermediate frames between two images.

It gives the impression that the first image has smoothly evolved into the second one.

It is an important method used in all types of animations.

In CSS3, Transforms (matrix, translate, rotate, scale) module can be used to achieve tweening.

94. 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.

95. Explain Scope and Scope Chain in javascript.?

Scope in JS, determines the accessibility of variables and functions at various

parts in one's code.

In general terms, the scope will let us know at a given part of code, what are the variables and functions that we can or cannot access.

There are three types of scopes in JS:

Global Scope

Local or Function Scope

Block Scope

Global Scope

Variables or functions declared in the global namespace have global scope, which means all the variables and functions having global scope can be accessed from anywhere inside the code

Function Scope

Any variables or functions declared inside a function have local/function scope, which means that all the variables and functions declared inside a function, can be accessed from within the function and not outside of it.

Block Scope

Block scope is related to the variables declared using let and const. Variables declared with var do not have block scope.

Block scope tells us that any variable declared inside a block { }, can be accessed only inside that block and cannot be accessed outside of it.

Scope Chain

JavaScript engine also uses Scope to find variables.

As you can see in the code above, if the javascript engine does not find the variable in local scope, it tries to check for the variable in the outer scope. If the variable does not exist in the outer scope, it tries to find the variable in the global scope.

If the variable is not found in the global space as well, reference error is thrown.

96. What is memoization?

Memoization is a form of caching where the return value of a function is cached based on its parameters. If the parameter of that function is not changed, the cached version of the function is returned.

97. What is the use of a constructor function in javascript?

Constructor functions are used to create objects in javascript.

When do we use constructor functions?

If we want to create multiple objects having similar properties and methods, constructor functions are used.

98. What are arrow functions?

Arrow functions were introduced in the ES6 version of javascript.

They provide us with a new and shorter syntax for declaring functions.

Arrow functions can only be used as a function expression.

Arrow functions are declared without the function keyword. If there is only one returning expression then we don't need to use the return keyword as well in an arrow function as shown in the example above. Also, for functions having just one line of code, curly braces { } can be omitted.

If the function takes in only one argument, then the parenthesis () around the parameter can be omitted as shown in the code above.

```
var obj1 = {
  valueOfThis: function(){
    return this;
  }
}
var obj2 = {
  valueOfThis: ()=>{
    return this;
  }
}

obj1.valueOfThis(); // Will return the object obj1
obj2.valueOfThis(); // Will return window/global object
```

The biggest difference between the traditional function expression and the arrow function, is the handling of the '**this'** keyword.

By general definition, the this keyword always refers to the object that is calling the function.

As you can see in the code above, **obj1.valueOfThis()** returns obj1, since **this** keyword refers to the object calling the function.

In the arrow functions, there is no binding of the **this** keyword.

The **this** keyword inside an arrow function, does not refer to the object calling it. It rather inherits its value from the parent scope which is the window object in this case.

Therefore, in the code above, **obj2.valueOfThis()** returns the window object.

99. What is the rest parameter and spread operator?

Both rest parameter and spread operator were introduced in the ES6 version of javascript.

Rest parameter (…)

It provides an improved way of handling parameters of a function

Using the rest parameter syntax, we can create functions that can take a variable number of arguments.

Any number of arguments will be converted into an array using the rest parameter.

It also helps in extracting all or some parts of the arguments.

Rest parameter can be used by applying three dots (...) before the parameters.

****Note- Rest parameter should always be used at the last parameter of a function:**

```
function extractingArgs(...args){
  return args[1];
}

// extractingArgs(8,9,1); // Returns 9

function addAllArgs(...args){
  let sumOfArgs = 0;
  let i = 0;
  while(i < args.length){
    sumOfArgs += args[i];
    i++;
  }
  return sumOfArgs;
}

addAllArgs(6, 5, 7, 99); // Returns 117
addAllArgs(1, 3, 4); // Returns 8
```

Spread operator (…)

Although the syntax of spread operator is exactly the same as the rest parameter, spread operator is used to spread an array, and object literals. We also use spread operators where one or more arguments are expected in a function call.

```
function addFourNumbers(num1,num2,num3,num4){
  return num1 + num2 + num3 + num4;
}

let fourNumbers = [5, 6, 7, 8];


addFourNumbers(...fourNumbers);
// Spreads [5,6,7,8] as 5,6,7,8

let array1 = [3, 4, 5, 6];
let clonedArray1 = [...array1];
// Spreads the array into 3,4,5,6
console.log(clonedArray1); // Outputs [3,4,5,6]


let obj1 = {x:'Hello', y:'Bye'};
let clonedObj1 = {...obj1}; // Spreads and clones obj1
console.log(obj1);

let obj2 = {z:'Yes', a:'No'};
let mergedObj = {...obj1, ...obj2}; // Spreads both the objects and merges it
console.log(mergedObj);
// Outputs {x:'Hello', y:'Bye',z:'Yes',a:'No'};
```

***Note- Key differences between rest parameter and spread operator:

- **Rest parameter is used to take a variable number of arguments and turns into an array while the spread operator takes an array or an object and spreads it**
- **Rest parameter is used in function declaration whereas the spread operator is used in function calls.**

100. What are classes in javascript?

Introduced in the ES6 version, classes are nothing but syntactic sugars for constructor functions.

They provide a new way of declaring constructor functions in javascript.

Below are the examples of how classes are declared and used:

```
// Before ES6 version, using constructor functions
function Student(name,rollNumber,grade,section){
  this.name = name;
  this.rollNumber = rollNumber;
  this.grade = grade;
  this.section = section;
}

// Way to add methods to a constructor function
Student.prototype.getDetails = function(){
  return 'Name: ${this.name}, Roll no: ${this.rollNumber}, Grade: ${this.grade}, Section:${this.section}';
}


let student1 = new Student("Vivek", 354, "6th", "A");
student1.getDetails();
// Returns Name: Vivek, Roll no:354, Grade: 6th, Section:A

// ES6 version classes
class Student{
  constructor(name,rollNumber,grade,section){
    this.name = name;
    this.rollNumber = rollNumber;
    this.grade = grade;
    this.section = section;
  }

  // Methods can be directly added inside the class
  getDetails(){
    return 'Name: ${this.name}, Roll no: ${this.rollNumber}, Grade:${this.grade}, Section:${this.section}';
  }
}

let student2 = new Student("Garry", 673, "7th", "C");
student2.getDetails();
// Returns Name: Garry, Roll no:673, Grade: 7th, Section:C
```

- Unlike functions, classes are not hoisted. A class cannot be used before it is declared.
- A class can inherit properties and methods from other classes by using the extend keyword.
- All the syntaxes inside the class must follow the strict mode(‘use strict’) of javascript. Error will be thrown if the strict mode rules are not followed.

101. What are generator functions?

Introduced in ES6 version, generator functions are a special class of functions. **They can be stopped midway and then continue from where it had stopped.** Generator functions are declared with the **function*** keyword instead of the normal **function** keyword:

```
function* genFunc(){
  // Perform operation
}
```

In normal functions, we use the **return** keyword to return a value and as soon as the return statement gets executed, the function execution stops:

```
function normalFunc(){
  return 22;
  console.log(2); // This line of code does not get executed
}
```

In the case of generator functions, when called, they do not execute the code, instead they return a **generator object** . This generator object handles the execution.

```
function* genFunc(){
  yield 3;
  yield 4;
}
genFunc(); // Returns Object [Generator] {}
```

The generator object consists of a method called **next()** , this method when called, executes the code until the nearest **yield** statement, and returns the yield value.

For example if we run the next() method on the above code:

```
genFunc().next(); // Returns {value: 3, done:false}
```

The **yield** keyword pauses generator function execution and the value of the expression following the **yield** keyword is returned to the generator's caller. It can be thought of as a generator-based version of the return keyword. **yield** can only be called directly from the generator function that contains it

As one can see the next method returns an object consisting of **value** and **done** properties.

Value property represents the yielded value.

Done property tells us whether the function code is finished or not. (Returns true if finished)

Generator functions are used to return iterators. Let’s see an example where an iterator is returned:

```
function* iteratorFunc() {
  let count = 0;
  for (let i = 0; i < 2; i++) {
       count++;
       yield i;
  }
  return count;
}

let iterator = iteratorFunc();
console.log(iterator.next()); // {value:0,done:false}
console.log(iterator.next()); // {value:1,done:false}
console.log(iterator.next()); // {value:2,done:true}
```

102. Explain WeakSet in javascript.

In javascript, Set is a collection of unique and ordered elements.

Just like Set, WeakSet is also a collection of unique and ordered elements with some key differences:

Weakset contains only objects and no other type.

An object inside the weakset is referenced weakly. This means, if the object inside the weakset does not have a reference, it will be garbage collected.

Unlike Set, WeakSet only has three methods, **add()** , **delete()** and **has()** .

```
const newSet = new Set([4, 5, 6, 7]);
console.log(newSet);// Outputs Set {4,5,6,7}

const newSet2 = new WeakSet([3, 4, 5]); //Throws an error


let obj1 = {message:"Hello world"};
const newSet3 = new WeakSet([obj1]);
console.log(newSet3.has(obj1)); // true
```

103. Explain WeakMap in javascript.

In javascript, Map is used to store key-value pairs. The key-value pairs can be of both primitive and non-primitive types.

WeakMap is similar to Map with key differences:

The keys and values in weakmap should always be an object.

If there are no references to the object, the object will be garbage collected.

```
const map1 = new Map();
map1.set('Value', 1);

const map2 = new WeakMap();
map2.set('Value', 2.3); // Throws an error

let obj = {name:"Vivek"};
const map3 = new WeakMap();
map3.set(obj, {age:23});
```

104. What is Polymorphism?

Polymorphism in Object-Oriented Programming is an ability to create a property, a function, or an object that has more than one realization.

Polymorphism is an ability to substitute classes that have common functionality in sense of methods and data. In other words, it is an ability of multiple object types to implement the same functionality that can work in a different way but supports a common interface.

105. What is Object Destructuring?

```
const classDetails = {
  strength: 78,
  benches: 39,
  blackBoard:1
}

const classStrength = classDetails.strength;
const classBenches = classDetails.benches;
const classBlackBoard = classDetails.blackBoard;
```

The same example using object destructuring:

```
const classDetails = {
  strength: 78,
  benches: 39,
  blackBoard:1
}

const {strength:classStrength, benches:classBenches,blackBoard:classBlackBoard} = classDetails;

console.log(classStrength); // Outputs 78
console.log(classBenches); // Outputs 39
console.log(classBlackBoard); // Outputs 1
```

As one can see, using object destructuring we have extracted all the elements inside an object in one line of code.

If we want our new variable to have the same name as the property of an object we can remove the colon:

```
const {strength:strength} = classDetails;
// The above line of code can be written as:
const {strength} = classDetails;
```

106. What are JavaScript Native Objects?

JavaScript has several built-in or native objects. These objects are accessible anywhere in your program and will work the same way in any browser running in any operating system.

Here is the list of JavaScript Native Objects −

- **JavaScript Number Object** − The Number object represents numerical date, either integers or floating-point numbers.
- **JavaScript Boolean Object** − The Boolean object represents two values, either "true" or "false".
- **JavaScript String Object** − The String object lets you work with a series of characters; it wraps Javascript's string primitive data type with a number of helper methods.
- **JavaScript Array Object** − The Array object lets you store multiple values in a single variable. It stores a fixed-size sequential collection of elements of the same type
- **JavaScript Date Object** − The Date object is a datatype built into the JavaScript language. Date objects are created with the new Date( ).
- **JavaScript Math Object** − The math object provides you properties and methods for mathematical constants and functions.
- **JavaScript RegExp Object** − The JavaScript RegExp class represents regular expressions, and both String and RegExp define methods that use regular expressions to perform powerful pattern-matching and search-and-replace functions on a text.

107. Explain what is Event Delegation?

Event delegation points to the process of using event propagation to handle events at a higher level in the DOM than the element on which the event originated. It enables you to avoid adding event listeners to particular nodes; instead, you can add a single event listener to a parent element.

108. Web worker and Service worker?

When executing scripts in an HTML page, the page becomes unresponsive until the script is finished.

A **web worker** is a JavaScript that runs in the background, independently of other scripts, without affecting the performance of the page. You can continue to do whatever you want: clicking, selecting things, etc., while the web worker runs in the background.

Web workers mimics the multi-threading model, allowing complex / data or processor intensive tasks to run in background. Ideal to keep your UI jank free when you have to deal with a lot of computations. Communication's with the webpage must go through web workers PostMessage method.

A **Service worker** is basically a script (JavaScript file) that runs in background and assists in offline first web application development. A Service worker can not directly interact with the webpage nor it can directly access the DOM as it runs on a different thread but can communicate with webpages through messages (Post Messages).

Service workers are designed to handle network requests and assist in offline first development, Push Notifications and background syncs. Communication's with the webpage must go through service workers PostMessage method.

109. Module.exports vs exports?

Setting `module.exports` allows the `database_module` function to be called like a function when `required`. Simply setting `exports` wouldn't allow the function to be

exported because node exports the object `module.exports` references. The following code wouldn't allow the user to call the function.

The following won't work.

```
exports = nano = function database_module(cfg) {return;}
```

The following will work if `module.exports` is set.

```
module.exports = exports = nano = function database_module(cfg) {return;}
```

110. What is Dependency Injection?

Dependency Injection is passing dependency to other **objects** or **framework**(dependency injector).

Dependency injection is basically providing the objects that an object needs (its dependencies) instead of having it construct them itself. It's a very useful technique for testing, since it allows dependencies to be mocked or stubbed out.

111. Different Types of SQL JOINs?

Here are the different types of the JOINs in SQL:

- (INNER) JOIN: Returns records that have matching values in both tables
- LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
- RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
- FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table

112. **Stored Procedures**

Stored Procedures are pre-compiled objects which are compiled for the first time and its compiled format is saved, which executes (compiled code) whenever it is called. For more about a stored procedure, please refer to the article Different types of Stored Procedure.

Functions

A function is compiled and executed every time whenever it is called. A function must return a value and cannot modify the data received as parameters. For more about functions, please refer to the article Different types of Functions.

Basic Differences between Stored Procedure and Function in SQL Server
The function must return a value but in Stored Procedure it is optional. Even a procedure can return zero or n values.

Functions can have only input parameters for it whereas Procedures can have input or output parameters.

Functions can be called from Procedure whereas Procedures cannot be called from a Function.

112. Types of Indexes in MySQL?

The types of indexes are:

1. Clustered: Clustered index sorts and stores the rows data of a table / view based on the order of clustered index key. Clustered index key is implemented in B-tree index structure.

2. Nonclustered: A non clustered index is created using clustered index. Each index row in the non clustered index has non clustered key value and a row locator. Locator positions to the data row in the clustered index that has key value.

3. Unique: Unique index ensures the availability of only non-duplicate values and therefore, every row is unique.

4. Full-text: It supports is efficient in searching words in string data. This type of indexes is used in certain database managers.

5. Spatial: It facilitates the ability for performing operations in efficient manner on spatial objects. To perform this, the column should be of geometry type.

6. Filtered: A non clustered index. Completely optimized for query data from a well defined subset of data. A filter is utilized to predicate a portion of rows in the table to be indexed.