**1.What are the primitive data types in JS?**

**In JavaScript, a** primitive (primitive value, primitive data type) is data that is not an object and has no methods or properties.

Ans. There are 7 primitive data types:

string.

number.

bigint.

boolean.

undefined.

symbol.

null.

**2. What's the difference between a variable that is: null, undefined or undeclared?**

Ans. Null is pointing to nothing in memory. Undefined is a variable that has not been assigned any value. Lastly,

undeclared is a variable that has not been properly declared using const, var, or let.

**3. What is the difference between while and do-while loops in JavaScript?**

Ans. Do / While VS While is a matter of when the condition is checked. A while loop checks the condition,

then executes the loop. A Do/While executes the loop and then checks the conditions.

**4. What language constructions do you use for iterating over object properties and array items?**

Ans. Language constructions which are used for iterating over object properties and array item are for loop, for..in, for each..in, map, reduce etc.

**5. What are the promises and how do they work?**

Ans. Promises are used to handle asynchronous operations in JavaScript. They are easy to manage when dealing with multiple asynchronous operations where

callbacks can create callback hell leading to unmanageable code.

Benefits of Promises-

Improves Code Readability

Better handling of asynchronous operations

Better flow of control definition in asynchronous logic

Better Error Handling

A Promise has four states:

fulfilled: Action related to the promise succeeded

rejected: Action related to the promise failed

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

settled: Promise has fulfilled or rejected.

**6.What are IIFEs and explain with an example where they can be used?**

Ans. An Immediately-invoked Function Expression (IIFE for friends) is a way to execute functions immediately, as soon as they are created. IIFEs are very useful

because they don't pollute the global object, and they are a simple way to isolate variables declarations.

Ex.- If you have 2 libraries that export an object with the same name, you can use IIFEs to ensure they don't conflict in your code. For example, the jQuery

and Cash JavaScript libraries both export $ as their main object.

**7. Explain event delegation.**

Ans. Event delegation refers to the process of using event propagation (bubbling) to handle events at a higher level in the DOM than the element

on which the event originated. It allows us to attach a single event listener for elements that exist now or in the future.

**8. Explain how this works in JavaScript.**

Ans. In JavaScript, the this keyword refers to an object. Which object depends on how this is being invoked (used or called). The this keyword refers to

different objects depending on how it is used: In an object method, this refers to the object.

Can you give an example of one of the ways that working with this has changed in ES6?

Ans. ES6 allows you to use arrow functions which uses the enclosing lexical scope.

**9. Explain how prototypal inheritance works.**

Ans. The Prototypal Inheritance is a feature in javascript used to add methods and properties in objects. It is a method by which an object can inherit

the properties and methods of another object. Traditionally, in order to get and set the [[Prototype]] of an object, we use Object. getPrototypeOf and Object.

**10. What is a closure, and how/why would you use one?**

Ans. 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.

Closures are useful because they let you associate data (the lexical environment) with a function that operates on that data. This has obvious parallels to

object-oriented programming, where objects allow you to associate data (the object's properties) with one or more methods.

**11. Can you describe the main difference between the** Array.forEach() loop and Array.map() methods and why you would pick one versus the other?

Ans. The forEach() method does not create a new array based on the given array. The map() method creates an entirely new array. The forEach() method returns “undefined“.

The map() method returns the newly created array according to the provided callback function.

**12. What's a typical use case for anonymous functions?**

Ans. Anonymous functions are often arguments being passed to higher-order functions, or used for constructing the result of a higher-order

function that needs to return a function.

If the function is only used once, or a limited number of times, an anonymous function may be syntactically lighter than using a named function.

**13. What's the difference between host objects and native objects?**

Ans. Host objects are supplied by other environment through a connection. The Host Objects are can not be same always because of the environment. Any object

which is not native is a host object, basically Host Objects are provided by the browser environment.

A Native Object is created by the developer using predefined classes of JavaScript. Native Objects are in your written script.

Than, a Custom Object is made by the developer from a custom (not predefined, or partially predefined) class.

**14. Explain the difference between: function Person(){}, var person = Person(), and var person = new Person()?**

Ans. function Person(){} is just a normal function declaration. The convention is to use PascalCase for functions that are intended to be used as constructors.

var person = Person() invokes the Person as a function, and not as a constructor. Invoking as such is a common mistake if the function is intended to be used as a constructor.

var person = new Person() creates an instance of the Person object using the new operator, which inherits from Person.prototype.

**15. Explain the differences on the usage of foo between function foo() {} and var foo = function() {}?**

Ans. function foo() {} is a Normal Function or Function Declaration and var foo = function() {} is a Anonymous Function or Expression Function or in simple words,

it may also be illustrated that it is variable inside which a function is wrapped up and thus that variable will no longer remain in a state of variable

it eventually became an accessible function itself.

**16. Can you explain what Function.call and Function.apply do? What's the notable difference between the two?**

Ans. call() Method: It calls the method, taking the owner object as argument. The keyword this refers to the ‘owner’ of the function

or the object it belongs to. We can call a method which can be used on different objects.

apply() Method: The apply() method is used to write methods, which can be used on different objects.

It is different from the function call() because it takes arguments as an array.

Difference: The call() method takes arguments separately. The apply() method takes arguments as an array. The apply() method is very handy

if you want to use an array instead of an argument list.

**17. Explain Function.prototype.bind.?**

Ans. prototype. bind() The bind() method creates a new function that, when called, has its this keyword set to the provided value,

with a given sequence of arguments preceding any provided when the new function is called.

**18. What's the difference between feature detection, feature inference, and using the UA string?**

Ans. These 3 are just practices of determining if a certain web technology feature exists in a user’s browser or environment.

Though features may vary with not just modern web technology but with programming languages as well.

Feature Detection

Feature detection is just a way of determining if a feature exists in certain browsers. A good example is a modern HTML5 feature ‘Location’.

Feature Inference

Feature Inference is when you have determined a feature exists and assumed the next web technology feature you are implementing unto your app exists as well.

Its usually bad practice to assume, so its better to explicitly specify features you want to detect and plan a fallback action.

UA String

UA String or User Agent String is a string text of data that each browsers send and can be access via navigator.userAgent.

These “string text of data” contains information of the browser environment you are targeting.

If you open your console and run "navigator.userAgent"

You’ll see it outputs a “string text of data” containing complete information of the environment you are currently using.

**19. Explain "hoisting"?**

Ans. Hoisting is the default behavior of moving all the declarations at the top of the scope before code execution. Basically,

it gives us an advantage that no matter where functions and variables are declared, they are moved to the top of their scope regardless

of whether their scope is global or local.

It allows us to call functions before even writing them in our code.

JavaScript only hoists declarations, not the initializations.

JavaScript allocates memory for all variables and functions defined in the program before execution.

**20. Describe event bubbling?**

Ans. Event bubbling is a type of event propagation where the event first triggers on the innermost target element, and then

successively triggers on the ancestors (parents) of the target element in the same nesting hierarchy till it reaches the outermost

DOM element or document object.

**21. Describe event capturing?**

Ans. Event capturing is one of two ways to do event propagation in the HTML DOM. In event capturing, an event propagates

from the outermost element to the target element. It is the opposite of event bubbling, where events propagate outwards

from the target to the outer elements. Capturing happens before bubbling.

**22. What's the difference between an "attribute" and a "property"?**

Ans. Attribute: Attributes are defined by HTML and are used to customize a tag.

Property: In contrast to the attributes, which are defined in

HTML, properties belong to the DOM. Since DOM is an object in JavaScript, we can get and set properties.

The attributes have a data type of string. So no matter the value of the attribute, it will always return a string.

Property: In contrast to the attributes, which are defined in HTML, properties belong to the DOM. Since DOM is an

object in JavaScript, we can get and set properties.

**23. What are the pros and cons of extending built-in JavaScript objects?**

Ans. The main argument against doing this is: if, in future, a browser decides to implement its own version of your method,

your method might get overridden (silently) and the browser's implementation (which is probably different from yours) would

take over. So not extending in the first place is future proofing your code.

On the flip side, if you decide to overwrite the browsers definition, any future developer working on your code won't know about

the change. They'll have a harder time getting up to speed.

**24. What is the difference between == and ===?**

Ans. == is used for comparison between two variables irrespective of the datatype of variable.

=== is used for comparision between two variables but this will check strict type, which means it will check datatype and compare two values.

**25. Explain the same-origin policy with regards to JavaScript?**

Ans. The same-origin policy is a critical security mechanism that restricts how a document or script loaded by one

origin can interact with a resource from another origin. It helps isolate potentially malicious documents, reducing possible attack vectors.

**26.Why is it called a Ternary operator, what does the word "Ternary" indicate?**

The ternary operator is an operator that exists in some programming languages, which takes three operands rather than the typical one or two that most operators use. It provides a way to shorten a simple if else block.

For example, consider the below JavaScript code.

variable\_name = (condition) ? value\_if\_true : value\_if false;

**27. What is strict mode? What are some of the advantages/disadvantages of using it?**

"use strict"; at the top of your code (or function), then the JS is evaluated in strict mode.

Strict mode throws more errors and disables some features in an effort to make your code more robust, readable, and accurate.

**28. What are some of the advantages/disadvantages of writing JavaScript code in a language that compiles to JavaScript?**

Advantages of JavaScript

Speed - JavaScript tends to be very fast because it is often run immediately within the client's browser. So long as it doesn't require outside resources, JavaScript isn't slowed down by calls to a backend server. Also, major browsers all support JIT (just in time) compilation for JavaScript, meaning that there's no need to compile the code before running it.

Simplicity - JavaScript's syntax was inspired by Java's and is relatively easy to learn compared to other popular languages like C++.

Popularity - JavaScript is everywhere on the web, and with the advent of Node.js, is increasingly used on the backend. There are countless resources to learn JavaScript. Both StackOverflow and GitHub show an increasing amount of projects that use JavaScript, and the traction it's gained in recent years is only expected to increase.

Interoperability - Unlike PHP or other scripting languages, JavaScript can be inserted into any web page. JavaScript can be used in many different kinds of applications because of support in other languages like Pearl and PHP.

Server Load - JavaScript is client-side, so it reduces the demand on servers overall, and simple applications may not need a server at all.

Rich interfaces - JavaScript can be used to create features like drag and drop and components such as sliders, all of which greatly enhance the user interface and experience of a site.

Extended Functionality - Developers can extend the functionality of web pages by writing snippets of JavaScript for third party add-ons like Greasemonkey.

Versatility - There are many ways to use JavaScript through Node.js servers. If you were to bootstrap Node.js with Express, use a document database like MongoDB, and use JavaScript on the frontend for clients, it is possible to develop an entire JavaScript app from front to back using only JavaScript.

Updates - Since the advent of ECMAScript 5 (the scripting specification that JavaScript relies on), ECMA International has been dedicated to updating JavaScript annually. So far, we have received browser support for ES6 in 2017 and look forward to ES7 being supported in the future.

Disadvantages of JavaScript

Client-Side Security - Since JavaScript code is executed on the client-side, bugs and oversights can sometimes be exploited for malicious purposes. Because of this, some people choose to disable JavaScript entirely.

Browser Support - While server-side scripts always produce the same output, different browsers sometimes interpret JavaScript code differently. These days the differences are minimal, and you shouldn't have to worry about it as long as you test your script in all major browsers.

**29. What tools and techniques do you use debugging JavaScript code?**

1. debugger

2. Display objects as a table

3. Try all the sizes

4. How to find your DOM elements quickly

5. Benchmark loops using console.time() and console.timeEnd()

**30. Explain the difference between mutable and immutable objects.**

a. What is an example of an immutable object in JavaScript?

Immutables are the objects whose state cannot be changed once the object is created. Strings and Numbers are Immutable.

b. What are the pros and cons of immutability?

Immutability has several advantages, including (but not limited to):

Programs with immutable objects are less complicated to think about, since you don't need to worry about how an object may evolve over time.

You don't need to make defensive copies of immutable objects when returning or passing to other functions, since there is no possibility an immutable object will be modified behind your back.

One copy of an object is just as good as another, so you can cache objects or re-use the same object multiple times.

Immutable objects are good for sharing information between threads in a multi-threaded environment since they don't need to be synchronized.

Operations on immutable objects return new immutable objects while operations that cause side-effects on mutable objects usually return void. This means several operations can be chained together. For instance

("foo" + "bar" + "baz").length()

In languages where functions are first class values, operations like map, reduce, filter, etc. are basic operations on collections. These can be combined in many ways, and can replace most loops in a program.

There are of course some disadvantages:

Cyclic data structures such as graphs are difficult to build. If you have two objects which can't be modified after initialization, how can you get them to point to each other?

Allocating lots and lots of small objects rather than modifying ones you already have can have a performance impact. Usually the complexity of either the allocator or the garbage collector depends on the number of objects on the heap.

Naive implementations of immutable data structures can result in extremely poor performance. For instance, concatenating many immutable strings (like in Java) is O(n2) when the best algorithm is O(n).

It is possible to write efficient immutable data structures, it just takes a little more thought.

c. How can you achieve immutability in your own code?

Mutable objects are those whose state is allowed to change over time. An immutable value is the exact opposite â€” after it has been created, it can never change.

Strings and Numbers are inherently immutable in javascript.

**31. Explain the difference between synchronous and asynchronous functions.**

The differences between asynchronous and synchronous include: Async is multi-thread, which means operations or programs can run in parallel. Sync is single-thread, so only one operation or program will run at a time.

Async is non-blocking, which means it will send multiple requests to a server.28

**32. What is an event loop?**

The event loop is the secret behind JavaScriptâ€™s asynchronous programming. JS executes all operations on a single thread, but using a few smart data structures, it gives us the illusion of multi-threading.

Letâ€™s take a look at what happens on the back-end.

a. What is the difference between call stack and task queue?

It has responsibility to see weather the call-stack is empty and does the task queue contains pending task to process.

If the call-stack is empty, it will push the task to the call-stack from the queue and the task gets processed.

**33. What are the differences between variables created using let, var or const?**

var keyword in JavaScript: The var is the oldest keyword to declare a variable in JavaScript.

Scope: Global scoped or function scoped. The scope of the var keyword is the global or function scope. It means variables defined outside the function can be accessed globally, and variables defined inside a particular function can be accessed within the function.

Example 1: Variable â€˜aâ€™ is declared globally. So, the scope of the variable â€˜aâ€™ is global, and it can be accessible everywhere in the program. The output shown is in the console.

<script>

var a = 10

function f(){

console.log(a)

}

f();

console.log(a);

</script>

Output:

10

10

let keyword in JavaScript: The let keyword is an improved version of the var keyword.

Scope: block scoped: The scope of a let variable is only block scoped. It canâ€™t be accessible outside the particular block ({block}). Letâ€™s see the below example.

Example 1: The output is shown in the console.

<script>

let a = 10;

function f() {

let b = 9

console.log(b);

console.log(a);

}

f();

</script>

Output:

9

10

const keyword in JavaScript: The const keyword has all the properties that are the same as the let keyword, except the user cannot update it.

Scope: block scoped: When users declare a const variable, they need to initialize it, otherwise, it returns an error. The user cannot update the const variable once it is declared.

Example 1: We are changing the value of the const variable so that it returns an error. The output is shown in the console.

<script>

const a = 10;

function f() {

a = 9

console.log(a)

}

f();

</script>

Output:

a=9

TypeError:Assignment to constant variable.

**34. What are the differences between ES6 class and ES5 function constructors?**

ES6 class constructors ES5 function constructors

As discussed above ES6 class constructors creates objects by adding function to their prototypes (Blueprint). ES5 function constructors also create objects along with inheritance property.

It ensures that this keyword used by the developer is referring to the object being created by the developer. Any function can be used as a function constructor and it primarily focuses on the creation of reusable object creation code.

Its syntax is similar to object creation in other object-oriented programming languages. Its syntax is unique and is not generally found in other object-oriented programming languages.

This can be said to be a syntax base for constructor functions and instantiate objects using a new operator. This also uses a new operator for object creation but focuses on how the objects are being instantiated.

**35. Can you offer a use case for the new arrow => function syntax?** How does this new syntax differ from other functions?

Arrow function â€” also called fat arrow functionâ€” is a new feature introduced in ES6 that is a more concise syntax for writing function expressions.

While both regular JavaScript functions and arrow functions work in a similar manner, there are certain differences between them.

// (param1, param2, paramN) => expression

// ES5

var add = function(x, y) {

return x + y;

};

// ES6

let add = (x, y) => { return x + y };

Arguments binding

Arrow functions do not have an arguments binding. However, they have access to the arguments object of the closest non-arrow parent function. Named and rest parameters are heavily relied upon to capture the arguments passed to arrow functions.

In case of a regular function:

let myFunc = {

showArgs(){

console.log(arguments);

}

};

myFunc.showArgs(1, 2, 3, 4);

36. What advantage is there for using the arrow syntax for a method in a constructor?

Arrow syntax binds this to the surrounding code which makes the code simpler and shorter.

The main advantage of using an arrow function as a method inside a constructor is that the value of this gets set at the time of the function creation and can't change after that. So, when the constructor is used to create a new object, this will always refer to that object.

37. What is the definition of a higher-order function?

Higher order functions are functions that operate on other functions, either by taking them as arguments or by returning them

**38.Can you give an example for destructuring an object or an array?**

let introduction = ["Hello", "I" , "am", "Sarah"];

let [greeting, pronoun] = introduction;

console.log(greeting);//"Hello"

console.log(pronoun);//"I"

let person = {name: "Sarah", country: "Nigeria", job: "Developer"};

let name = person.name;

let country = person.country;

let job = person.job;

console.log(name);//"Sarah"

console.log(country);//"Nigeria"

console.log(job);//Developer"

**39. Can you give an example of generating a string with ES6 Template Literals?**

var name = "Brendan";

console.log(`Yo, ${name}!`);

// => "Yo, Brendan!"

**40.Can you give an example of a curry function and why this syntax offers an advantage?**

Currying is a pattern where a function with more than one parameter is broken into multiple functions that, when called in series, will accumulate all of the required parameters one at a time.

There is a lot of mathematical and computer science theory behind currying in functional programming and I encourage you to read more on Wikipedia. Interesting fact: Currying is named after a mathematician Haskell Curry, not the food.

Basic example

Most basic example, that you will find everywhere, is this:

const sum = x => y => x + y;

// returns the number 3

sum (2)(1);

// returns a function y => 2 + y

sum (2);

**41. What are the benefits of using spread syntax and how is it different from rest syntax?**

The difference is that spread syntax spreads the elements of an array or object into individual variables, whereas rest syntax condenses individual variables and puts them into an array or object.

function myBio(firstName, lastName, company) {

return `${firstName} ${lastName} runs ${company}`;

}

// Use spread to expand an arrayâ€™s items into individual arguments:

myBio(...["Oluwatobi", "Sofela", "CodeSweetly"]);

// The invocation above will return:

â€œOluwatobi Sofela runs CodeSweetlyâ€

Now, the rest ðŸ˜. The rest syntax follows the same convention as the spread syntax using the three dots before that which we choose to copy. But, the difference between the rest and spread syntax is that while spread copies everything, rest is used when we want to retrieve all remaining elements (or all existing elements) after a destructuring operation.

const notSoDomesticAnimals = ['Salamander', 'Iguana', 'Moth', 'Sloth'];

const [salamander, iguana, â€¦rest] = notSoDomesticAnimals;

42.How can you share code between files?

Share JavaScript Code

Create JavaScript files that export code in the same folder as the component importing the code. Import the code using a relative path. ...

Create a service component (library), which is a component folder that contains one or more JavaScript files that export code.

**43. Why you might want to create static class members?**

The advantage of using a static class is that the compiler can check to make sure that no instance members are accidentally added. The compiler will guarantee that instances of this class cannot be created.

Static classes are sealed and therefore cannot be inherited. They cannot inherit from any class except Object.