**Variables and values**

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

<https://stackoverflow.com/questions/359494/which-equals-operator-vs-should-be-used-in-javascript-comparisons>

* **How do var, const and let keyword differ?**
* **What is hoisting?**
* **Are functions hoisted? Are function expressions hoisted?**
* **Few questions on hoisting:**

[https://www.freecodecamp.org/news/function-hoisting-hoisting-interview-questions- b6f91dbc2be8/](https://www.freecodecamp.org/news/function-hoisting-hoisting-interview-questions-%20%20%20%20%20%20%20b6f91dbc2be8/)

<https://medium.com/@pvivek4/hoisting-demystified-with-popular-interview-questions-38a93ea441ff>

* **What is scope? Is JS a function scoped or a block scoped language or both?**

<https://www.w3schools.com/js/js_scope.asp>

Scope determines the accessibility (visibility) of variables. In JavaScript, objects and functions are also variables. Scope determines the accessibility of variables, objects, and functions from different parts of the code.

JavaScript has 3 types of scope:

1. Block scope - let and const help to create block scope. Variables declared with the var keyword can NOT have block scope.

{  
  let x = 2;  
}  
// x can NOT be used here

1. Function scope - Variables declared within a JavaScript function, become LOCAL to the function. Local variables have Function Scope: They can only be accessed from within the function. Variables declared with var, let and const are quite similar when declared inside a function.

function myFunction() {  
  var carName = "Volvo";   // Function Scope

let carName = "Volvo";   // Function Scope

const carName = "Volvo"; // Function Scope  
}

1. Global scope - A variable declared outside a function, becomes GLOBAL.

Variables declared Globally (outside any function) have Global Scope. Global variables can be accessed from anywhere in a JavaScript program. Variables declared with var, let and const are quite similar when declared outside a block. They all have Global Scope:

var x = 2;       // Global scope

let x = 2;       // Global scope

const x = 2;       // Global scope

* **What is a closure? How closures work? What is the relation between scope and closure?**

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 ***functions*** that ***refer to*** independent ***(free) variables***. In other words, the function defined in the closure ***‘remembers’ the environment in which it was created***.

Note: Free variables are variables that are neither locally declared nor passed as parameter.

<https://www.freecodecamp.org/news/lets-learn-javascript-closures-66feb44f6a44/#:~:text=function%20sayHello()%20%7B%20var%20say>,'

<https://stackoverflow.com/questions/111102/how-do-javascript-closures-work?rq=1>

<https://stackoverflow.com/questions/750486/javascript-closure-inside-loops-simple-practical-example?rq=1>

function numberGenerator() {

// Local “free” variable that ends up within the closure

var num = 1;

function checkNumber() {

console.log(num);

}

num++;

return checkNumber;

}

var number = numberGenerator();

number(); // 2

* **What are the uses or applications of closure?**
* **Explain variable capture (captured variables) in Javascript.**

<https://stackoverflow.com/questions/10095404/understanding-variable-capture-by-closures-in-javascript-node>

**Primitive values**

* **What are the primitive types in JS?**
* **Give example of reference types in JS.**
* **How are primitive types different from reference types?**
* **Is string a reference type or value type in JS?**

It is a primitive type.

* **What is the use of Symbol primitive type?**

**Control flow and data flow**

* **What the difference is between spread and rest operator?**
* **Write code to demonstrate spread operator?**
* **What are you going to make an object iterable so that you can use constructs such as for-of loop?**
* **What is an arrow function? How does it differ from normal functions? How value of this is decided inside arrow function?**

<https://www.w3schools.com/js/js_arrow_function.asp>

* **What are the difference between function and function expression?**
* **What are generator functions and how does it differ from normal functions?**
* **Is function an object? Explain how?**
* **Describe the properties of an anonymous function in JavaScript?**

**Modularity**

* **What is an IIFE? When should it be used?**
* **What is a module? How do you create modules in JS?** <https://www.w3schools.com/js/js_modules.asp>
* **What are the advantages of ES6 modules? Or Why should you create a module?**
* **What is 'this' keyword in JavaScript? How does the "this" keyword work, and when should it be used?**

<https://stackoverflow.com/questions/3127429/how-does-the-this-keyword-work-and-when-should-it-be-used>

* **Few questions on this keyword**

<https://dmitripavlutin.com/javascript-this-interview-questions/>

* **What are the rules to determine value of this?**
* **Can you explain how .apply, .call, .bind is available to any function?**
* **How .call, .apply and .bind function differ from one another? When should they be used? Explain with example code?**
* **What is a prototype?**

<https://www.w3schools.com/js/js_object_prototypes.asp>

* **What is \_\_proto\_\_ (dandur proto)? When should it be used?**
* **What is the difference between classical and prototypical inheritance?**
* **How do you implement encapsulation in JS?**
* **What are the different ways of reusing code in JS?**

**Collections**

* **How do you implement HashMap in JS?**
* **What are the differences between for-of, for-in, foreach and for loop? Which one should be used and under what circumstances?**
* **What is object destructuring?**

**Asynchronicity**

* **What is a promise? What problems does promise solve?**
* **What is the use of async and await?**

<https://www.w3schools.com/js/js_async.asp>

* **What is a JS/Browser event loop?**
* **Explain the working of timers (setTimeout, setInterval and clearInterval) in JavaScript? Also elucidate the drawbacks of using the timer, if any?**
* **What is the output of below code?**

setTimeout(()=> {

console.log("Hello");

}, 0);

console.log("World");

**Output**: World Hello

* **What is micro and macro task in JavaScript?**
* **What is the difference between microtasks and tasks in JavaScript?**
* **Why promise is a microtask?**
* **What is Promise vs async?**

**JavaScript HTML DOM**

* **What is a DOM in JavaScript?**
* **How does a DOM model work?**
* **How DOM is created in JavaScript?**
* **What is benefit of DOM in JavaScript?**
* **What are the 3 parts of DOM?**
* **What is the main purpose of DOM?**

**WebWorker**

**Miscellaneous topics**

* **What is JavaScript? Is it synchronous or asynchronous language?**

JavaScript is the **synchronous, single-threaded language** but with the help of event-loop and promises, JavaScript is used to do asynchronous programming. That just means that only one operation can be in progress at a time.

JavaScript is a single-threaded, non-blocking, asynchronous, concurrent programming language with lots of flexibility.

* **Is JS an interpreted or a compiler based language or is it both?**

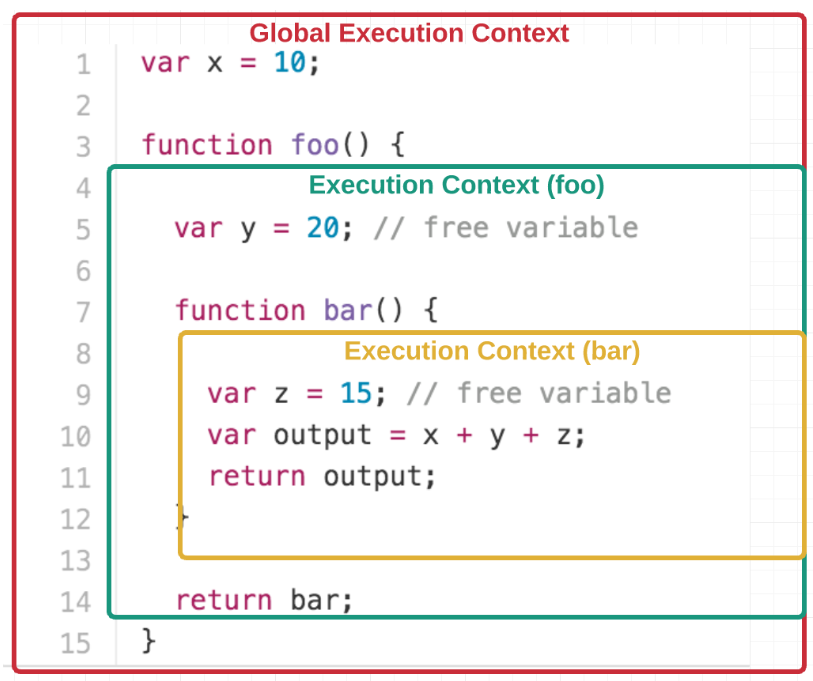
JavaScript (often shortened to JS) is a lightweight, interpreted, object-oriented language with first-class functions, and is best known as the scripting language for Web pages, but it's used in many non-browser environments as well.

* **Is JS an object oriented language or functional language or both?**

JS is both object oriented and functional language.

* **How are JavaScript, ECMAScript and TypeScript related?**
* **Is GC manual or automatic in JS?**
* **How garbage collection happens in JS?**
* **How GC happens in JS? Is GC automatic in JS?**
* **What is execution context? Why JS is single threaded?**

Execution context is an abstract concept used by the ECMAScript specification to track the runtime evaluation of code. This can be the global context in which your code is first executed or when the flow of execution enters a function body.



At any point in time, there can only be one execution context running. That’s why JavaScript is “single threaded,” meaning only one command can be processed at a time.

Typically, browsers maintain this execution context using a “stack.” A stack is a Last In First Out (LIFO) data structure, meaning the last thing that you pushed onto the stack is the first thing that gets popped off it. (This is because we can only insert or delete elements at the top of the stack.)

* **What are mixins? Why are they important?**
* **What is the most efficient way to deep clone an object in JavaScript?**
* **What is event bubbling?**

UserDetailsService

lifecycle management

spring JPA

custom error page

exception handling

logging - log4j2, logging using lombok- https://howtodoinjava.com/spring-boot2/logging/logging-with-lombok/