What is JavaScript and what is it used for?

JavaScript is a text-based programming language used both on the client-side and server-side that allows you to make web pages interactive. Where HTML and CSS are languages that give structure and style to web pages, JavaScript gives web pages interactive elements that engage a user.

JavaScript improves the user experience of the web page by converting it from a static page into an interactive one, JavaScript adds **behavior** to web pages.

JavaScript is mainly **used** for web-based applications and web browsers. But JavaScript is also used beyond the Web in software, servers and embedded hardware controls.

* Adding interactive behavior to web pages

### Creating web and mobile apps

### Building web servers and developing server applications

### Game development

### What is the difference between let, var, and const in JavaScript?

Javascript provides us with multiple ways to declare a variable, but which one should we use.

Suppose we created a function and we need a constant variable inside the function, which means we want a variable that will not be updated in any case. Here if we will declare a var variable or let variable, then it can be updated, but if we declare a const variable, it will not be updated in any case and will work fine with our function.

### In [JavaScript](https://www.geeksforgeeks.org/javascript-tutorial/), users can declare a variable using 3 keywords that are [var, let, and const](https://www.geeksforgeeks.org/how-to-declare-variables-in-different-ways-in-javascript/).

[**JavaScript var keyword**](https://www.geeksforgeeks.org/var-keyword-in-java/)**:** The *var* is the oldest keyword to declare a variable in JavaScript.

**Scope:**[Global scoped](https://www.geeksforgeeks.org/understanding-variable-scopes-in-javascript/#:~:text=types%20of%20scopes-,Global%20Scope,-%E2%80%93%20Scope%20outside%20the) 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.

### [**JavaScript let keyword:**](https://www.geeksforgeeks.org/javascript-let/)The *let*keyword is an improved version of the *var* keyword.

### **Scope:**[**block scoped:**](https://www.geeksforgeeks.org/javascript-es2015-block-scoping/) The scope of a *let*variable is only block scoped. It can’t be accessible outside the particular block ({block}).

[**const**](https://www.geeksforgeeks.org/javascript-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:](https://www.geeksforgeeks.org/javascript-es2015-block-scoping/) 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.

### Can you explain closure in JavaScript?

This pattern of public, private, and privileged members is possible because JavaScript has **closures**. What this means is that an inner function always has access to the vars and parameters of its outer function, even after the outer function has returned.

Private and privileged members can only be made when an object is constructed. Public members can be added at any time.

What is the difference between == and === in JavaScript?

Double equals (==) is a comparison operator, which transforms the operands having the same type before comparison.

So, when you compare string with a number, JavaScript converts any string to a number. An empty string is always converts to zero. A string with no numeric value is converts to NaN (Not a Number), which returns false

=== (Triple equals) is a strict equality comparison operator in JavaScript, which returns false for the values which are not of a similar type. This operator performs type casting for equality. If we compare 2 with “2” using ===, then it will return a false value.

### How do you declare a function in JavaScript?

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is executed when "something" invokes it (calls it).

A JavaScript function is defined with the function keyword, followed by a **name**, followed by parentheses **()**.

Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

The parentheses may include parameter names separated by commas:  
**(parameter1, parameter2, ...)**

### What is the difference between null and undefined in JavaScript?

**Null:** It is the intentional absence of the value. It is one of the primitive values of JavaScript.

**Undefined:** It means the value does not exist in the compiler. It is the global object.

### Can you explain the prototype in JavaScript and its purpose?

### JavaScript is a prototype based language, so, whenever we create a function using JavaScript, JavaScript engine adds a *prototype* property inside a function, Prototype property is basically an object (also known as Prototype object), where we can attach methods and properties in a prototype object, which enables all the other objects to inherit these methods and properties.

### prototype are only intended to provide methods for the primitive values, but the wrapper objects should not be directly constructed.

### How would you implement inheritance in JavaScript?

### Classes are inherited using the extends keyword. The class that is being inherited is known as the parent class, and the class that is inheriting the parent class is known as the child class. Super keyword in javascript is used to invoke the methods of the parent class.

### Can you explain event bubbling in JavaScript?

Event bubbling directs an event to its target. It works like this: When an element (like a button) is clicked, an event is directed to the element. If an event handler is set for the element, the event handler is triggered. Then the event "bubbles up" to the elements parent.

### What is the difference between synchronous and asynchronous code in JavaScript?

### Synchronous means the code runs in a particular sequence of instructions given in the program

### whereas asynchronous code execution allows to execution of the upcoming instructions immediately.

### Synchronous processing uses only one thread where it executes all operations in succession.

### In contrast, each asynchronous operation is done in a different thread that reports back to the main thread with the result when complete or with an error in case of failure, leaving that thread open to process other requests.

### Can you explain the use of event delegation in JavaScript?

### Event Delegation is basically a pattern to handle events efficiently. Instead of adding an event listener to each and every similar element, we can add an event listener to a parent element and call an event on a particular target using the .target property of the event object.

### How would you handle exceptions in JavaScript?

### Primary method of dealing with exceptions in JavaScript is the try-catch. In a nutshell, the try-catch is a code block that can be used to deal with thrown exceptions without interrupting program execution.

### How would you handle a large data set with memory constraints in JavaScript?

### JavaScript automatically allocates memory when objects are created and frees it when they are not used anymore

1. Allocate the memory you need
2. Use the allocated memory (read, write)
3. Release the allocated memory when it is not needed anymore

Can you explain the concept of currying in JavaScript?

It is a technique in functional programming, transformation of the function of multiple arguments into several functions of a single argument in sequence.

We simply wrap a function inside a function, which means we are going to return a function from another function to obtain this kind of translation. The parent function takes the first provided argument and returns the function that takes the next argument and this keeps on repeating till the number of arguments ends. Hopefully, the function that receives the last argument returns the expected result.