Memory management in JavaScript is handled automatically by the runtime environment, typically the JavaScript engine in web browsers or Node.js. JavaScript uses a garbage collector to manage memory and ensure that developers do not need to manually allocate or deallocate memory.

Memory Life Cycle

Irrespective of the programming language, the memory life cycle follows the following stages:

1. Allocates the memory we need: JavaScript allocates memory to the object created.
2. Use the allocated memory.
3. Release the memory when not in use: Once the allocated memory is released, it is used for other purposes. It is handled by a JavaScript engine.

The second stage is the same for all the languages. However, the first and last stages are implicit in high-level languages like JavaScript.

JavaScript engines have two places to store data

- **Stack**: It is a data structure used to store static data. Static data refers to data whose size is known by the engine during compile time. In JavaScript, static data includes primitive values like strings, numbers, boolean, null, and undefined. References that point to objects and functions are also included. A fixed amount of memory is allocated for static data. This process is known as static memory allocation.

- **Heap**: It is used to store objects and functions in JavaScript. The engine doesn’t allocate a fixed amount of memory. Instead, it allocates more space as required.

Differences between stack and heap

Stack:
1. Primitive data types and references
2. Size is known at compile time
3. Fixed memory allocated

Heap:
1. Objects and functions
2. Size is known at run time
3. No limit for object memory