Garbage Collection

A **garbage collector** automatically frees up memory that a program isn't using anymore.

For example, say we did this in JavaScript:

```JavaScript
function getMin(nums) {
    // NOTE: this is NOT the fastest
    // way to get the min!

    var numsSorted = nums.sort();
    return numsSorted[0];
}

var myNums = [5, 3, 1, 4, 6];
console.log(getMin(myNums));
```

Look at `numsSorted` in `getMin()`. We allocate that whole array inside our function, and once the function returns we don't need the array anymore. In fact, once the function returns we *don't have any references to it anymore*!

What happens to that array in memory? The JavaScript garbage collector will notice we don't need it anymore and free up that space.

In some lower-level languages, like C, we don't have a garbage collector. So we need to manually free up any memory we're not using anymore:

```c
// make a string that can hold 15 characters
// including the terminating null byte ('\0')
str = malloc(15);

// ... do some stuff with it ...

// we're done. free that memory!
free(str);
```

We sometimes call this **manual memory management**.

Some languages, like C++, have both manual and automatic memory management.

What's next?

If you're ready to start applying these concepts to some problems, check out our mock coding interview questions (/next).

They mimic a real interview by offering hints when you're stuck or you're missing an optimization.

Try some questions now ➜

Want more coding interview help?

Check out **interviewcake.com** for more advice, guides, and practice questions.