# What is “Space Complexity”?

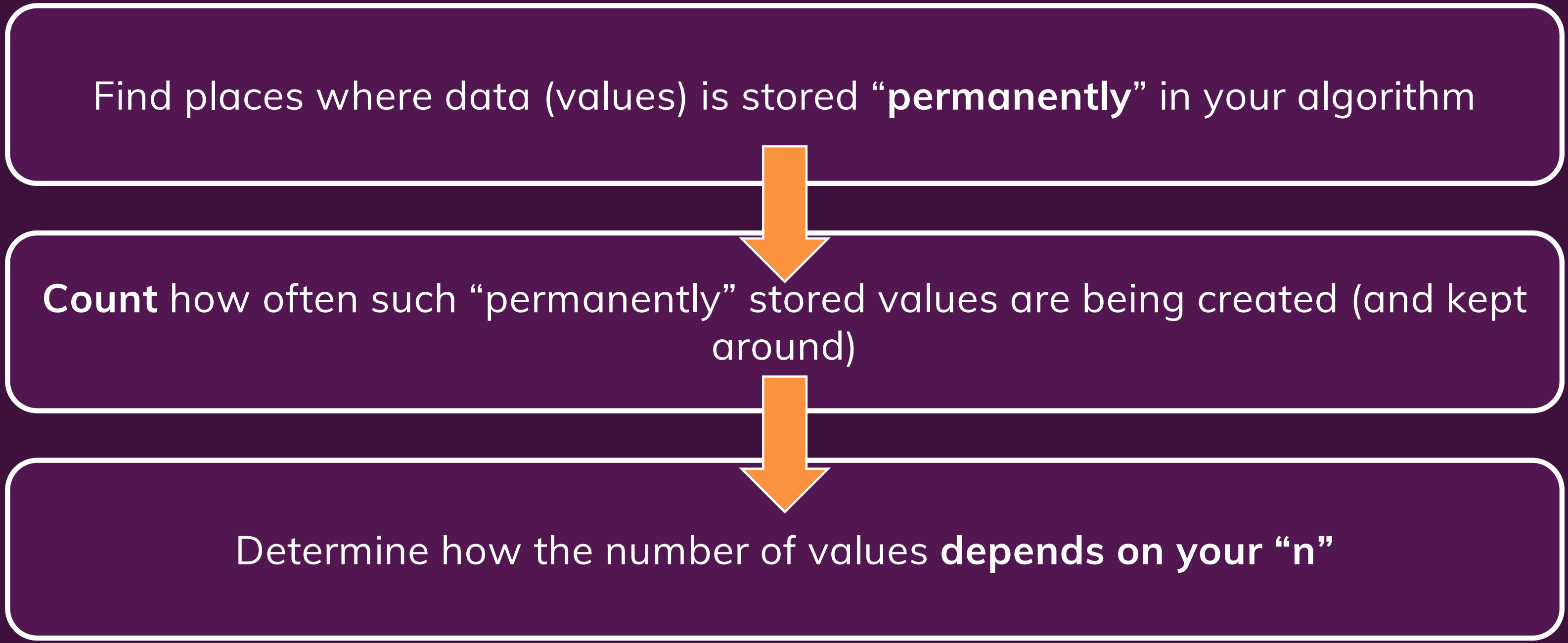
How much **space in memory** does your algorithm occupy?

All values in JavaScript are stored in memory

Especially arrays and objects can take up a bit more space

Generally, in JavaScript, **you won’t need to worry** about space complexity and memory too much though

# Deriving Space Complexity



Find places where data (values) is stored “**permanently**” in your algorithm

**Count** how often such “permanently” stored values are being created (and kept around)

Determine how the number of values **depends on your “n”**

➔ O(n), O(1) etc. exists for space complexity as well

# Examples

| Algorithm | Space Complexity | Reason |
|---|---|---|
| Factorial (Loop) | O(1) | We operate on **one and the same** number, **no** new (“permanent”) value is created per iteration |
| Factorial (Recursive) | O(n) | A **new value is created for every nested function call** (the parameter received) |
| Linear Search | O(1) | **No** new ”permanent” values are created during the iteration |
| Binary Search | O(1) | **No** new ”permanent” values are created during the iteration |

# More Examples

| Algorithm | Space Complexity | Reason |
|---|---|---|
| Bubble Sort | O(1) | **No** new ”permanent” values are created during the iteration |
| Quicksort | O(n)<br>*(O(log n) is possible)* | **Nested function calls** with new values being created |
| Merge Sort | O(n) | **Nested function calls** with new values being created |