**Big O Cheatsheet**

<https://zerotomastery.io/cheatsheets/big-o-cheat-sheet/?utm_source=udemy&utm_medium=coursecontent>

**What is Good Code?**

1. Readable
2. Scalable (Big O)

When we talk about BigO and scalability of code, we simply mean when we grow bigger and bigger with

our input, how much does the algorithm or function slow down? If the list of characters, let's say elements here, so that is Nemo, so characters in Finding Nemo in our era as that increases.

How many more operations do we have to do? That's all it is.

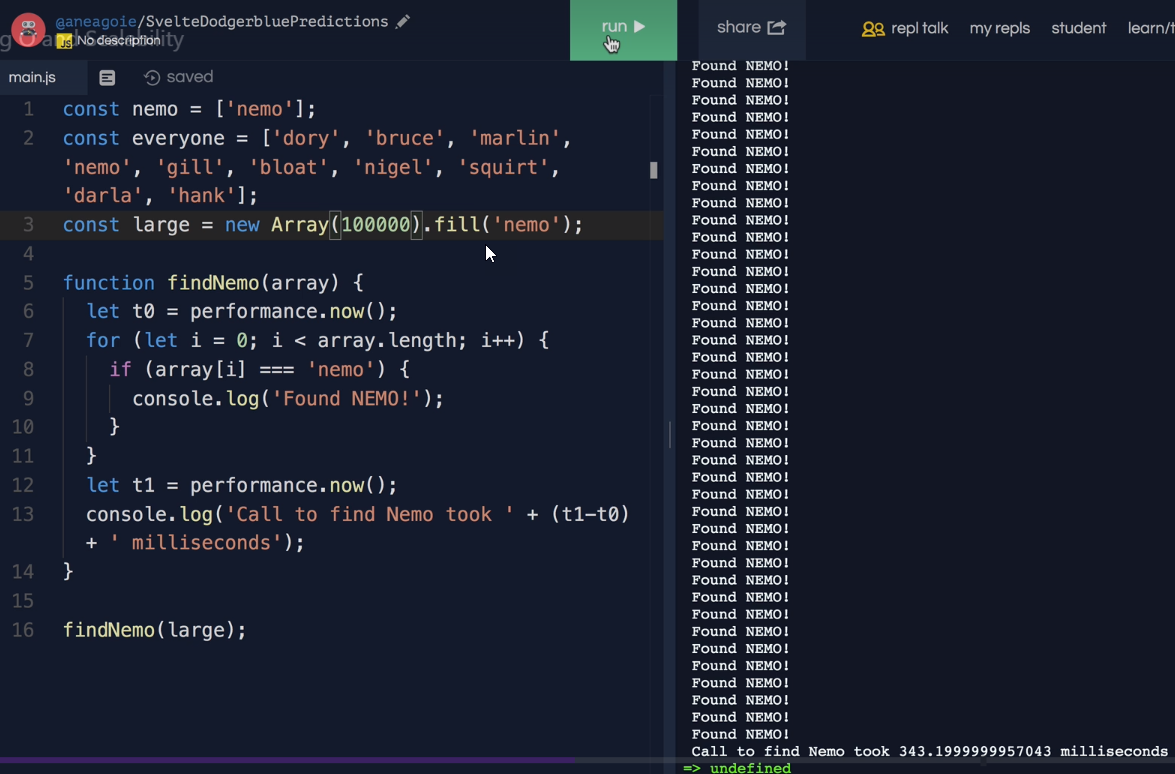
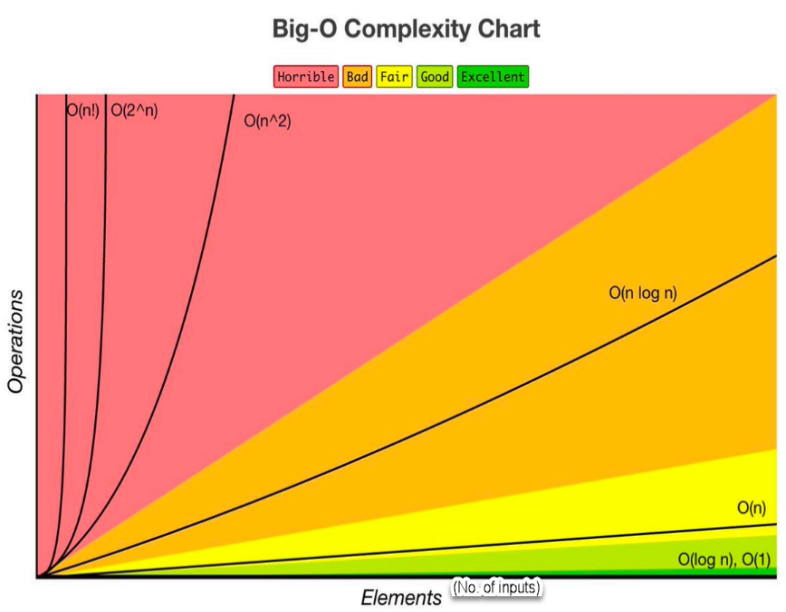
This is what we call algorithmic efficiency.

When we talk about Big O and scalability of code, we simply mean when we grow bigger and bigger with our input, how much does the algorithm slow down? The less it slows down or the slower it slows down, the better it is.

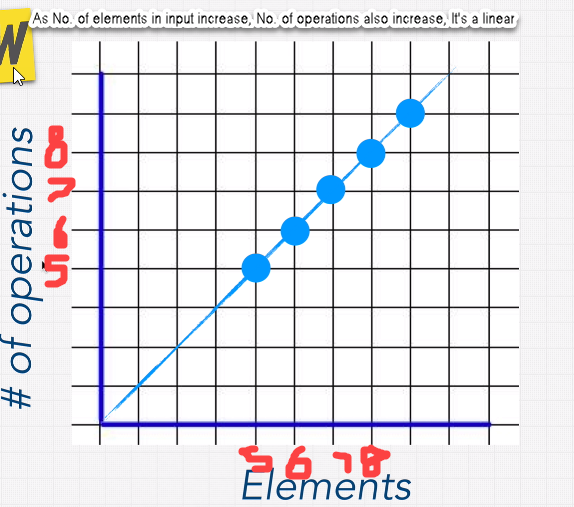
So instead of using performance done now and using time to measure the efficiency of our function, we can just calculate how many operations (example comparison & incremental operation in for loop) a computer has to perform because each operation takes time on a computer.

So BigO allows us and concerns us with how many steps it takes in a function.

Big O notation is the language we use for talking about how long an algorithm takes to run. We can compare to different algorithms or in this case functions using BigO and say which one is better than the other when it comes to scale, regardless of our computer differences. And we can measure BigO like below graph.

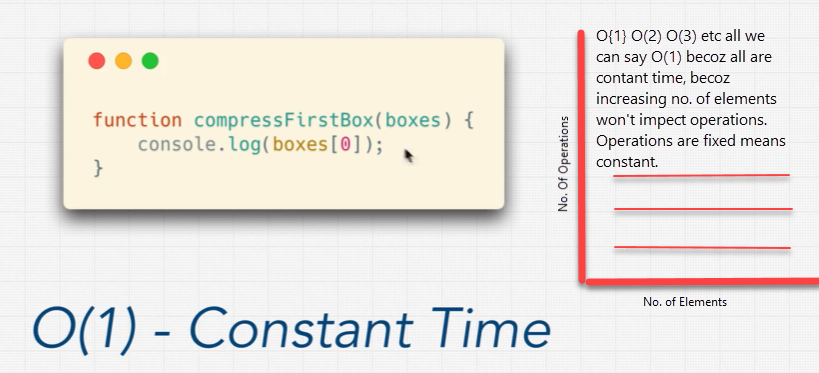


**O(N)**

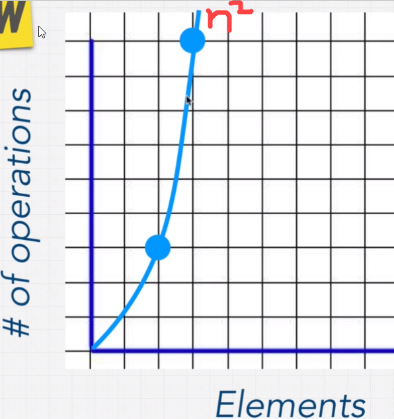
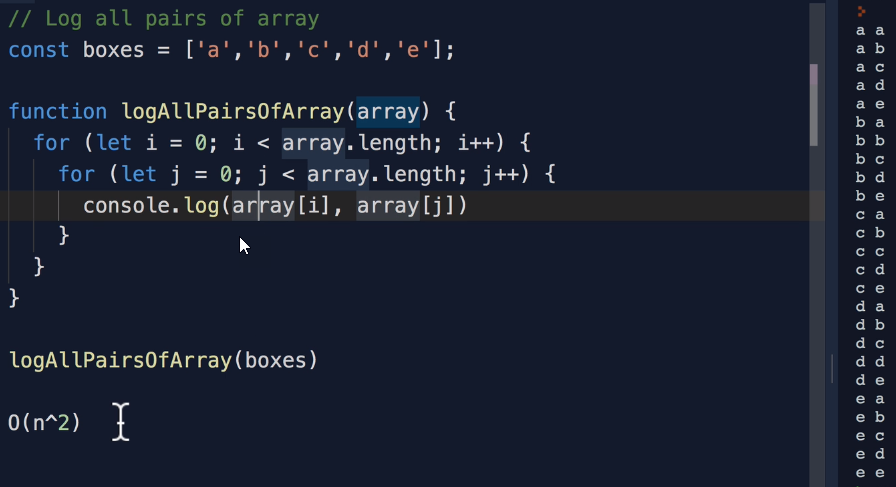


If we have five items in the array. It's going to be five operations, five loops. Six is the same. Six items is six operations, seven is seven operations and eight is eight operations. Do we see a little bit of a pattern here? Well, we can draw a line through it. This is linear, right, as our number of inputs increase, the number of operations increased as well. And here, ladies and gentlemen, we've learned our very first bigO notation. We say that the Finding Nemo function has a big notation of O(n). 🡪 “n” is an arbitrary number, we can use anything, but “n” is a standard.

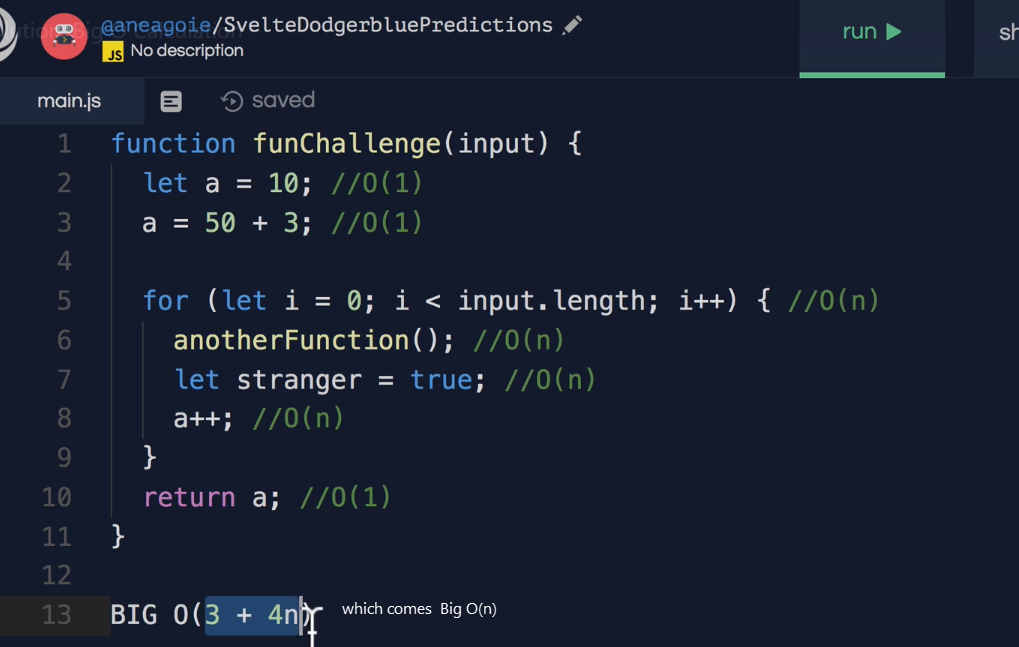
**O(1)**

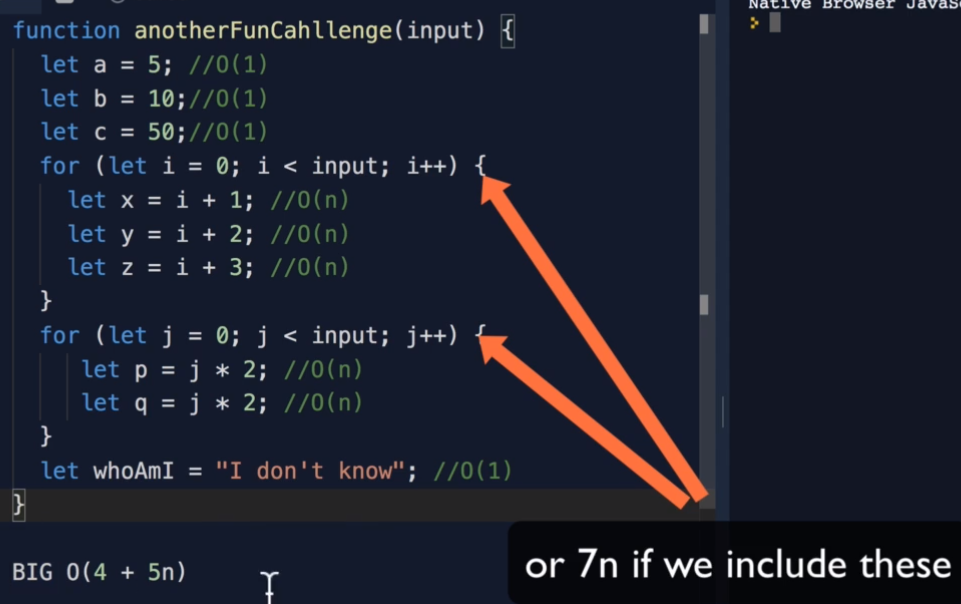


**O(n^2)**



Exercise





**Rule Book**

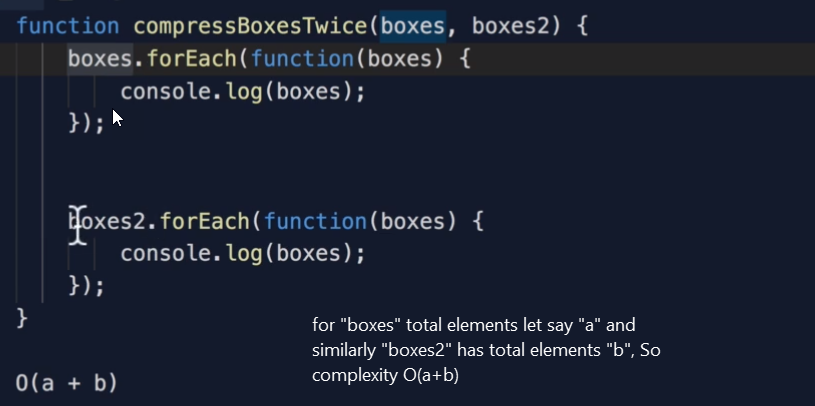
**Rule1: Worst Case**



**Rule2: Remove Constant**

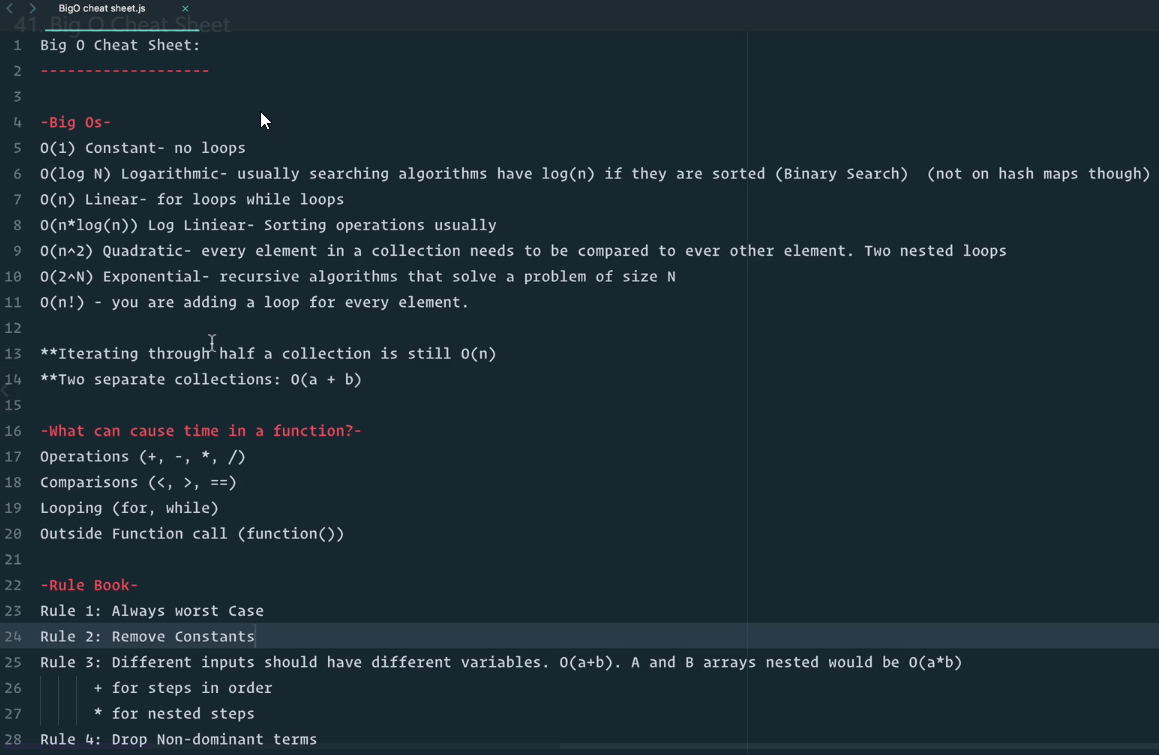
If time complexity calculated as O(2n) or O(n+5) etc then make it O(n)

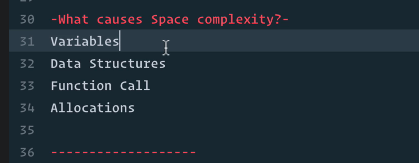
**Rule3: Different Terms for Input**



**Rule4: Drop Non-Dominants**

If complexity O(n^2 + n) then drop n which is smaller then n^2 So complexity is Big O(n^2)





**Note:** [**http://bigocheatsheet.com**](http://bigocheatsheet.com)

