# Estructuras de datos y algoritmos

## ¿Qué es buen código?

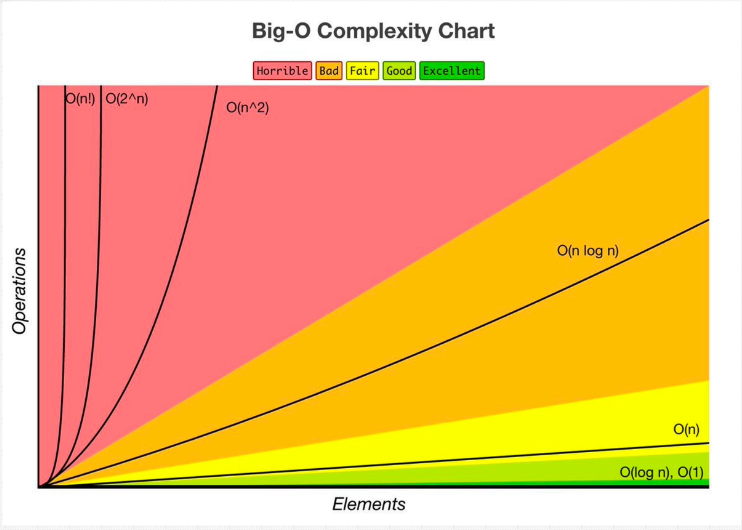
Es el código:

1. Legible
2. Escalable

## ¿Que causa tiempo en una función?

1. Operaciones (+, -, \*, /)
2. Comparativas (<, >, =)
3. Looping (for, while)
4. Llamada a función externa (function())

## Cuadro de complejidad Big O



## Reglas Big O

### Regla 1: Peor caso

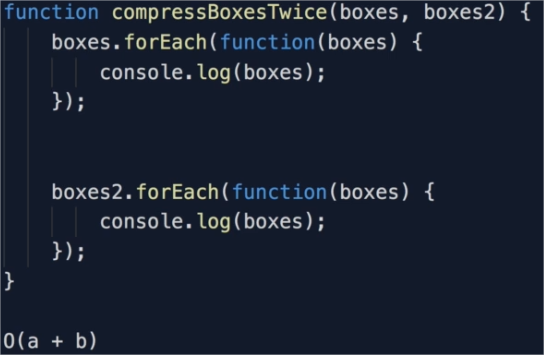
Siempre se calcula en base al peor caso.

### Regla 2: Remueve las constantes

A la hora de expresar el resultado, no es necesario ser tan especifico, solo hay que mostrar el tipo de Big O que sería. Por ejemplo, tenemos O(n/2 + 101), esto se expresa como O(n). A esto se le dice remover las constantes.

### Regla 3: Términos diferentes para inputs

Significa utilizar diferentes términos para diferentes inputs. Por ejemplo, teniendo en cuenta que boxes = a y boxes2 = b.



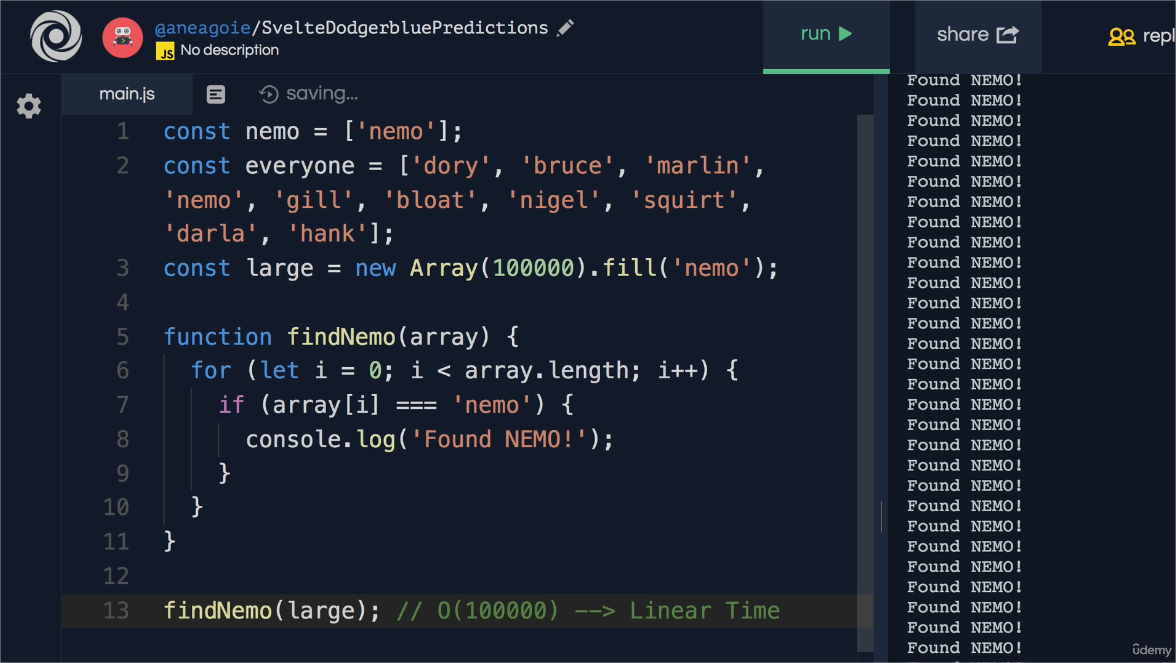
## Regla 4: Elimina los no dominantes

Significa preocuparse por el termino más importante y eliminar el menos importante, es decir, se elimina el menos dominante.

Por ejemplo, cuando una función sea O(n + n^2), aplicando esta regla queda como O(n^2)

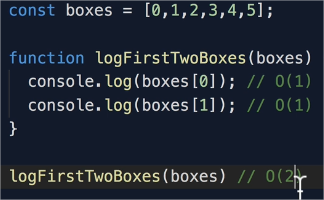
## O(n)

Es la notación linear/tiempo linear. Significa que el Big O depende de (n). Por ejemplo:



Es decir, el número de operaciones aumenta a medida que aumenta **n**.

## O(1)

Es la notación constante/tiempo constante. Significa que siempre será una y solo una operación. Por ejemplo:  


A pesar de que se ejecuten dos operaciones y sea O(2), esto se describiría como O(1).

## O(n^2)

Es la notación cuadrática. Significa que el número de operaciones incrementa en proporción al cuadrado de n.

