Curso de Closures y Scope en JavaScript

# Qué es el Scope y cómo funciona el Global Scope

Scope: Es el alcance que va a tener una variable dentro del código. En otras palabras, el Scope se encargará de decidir a qué bloques de código va a acceder una variable.

**Global Scope** : No están dentro de funciones o bloques, por lo tanto se puede acceder a ellas de manera global.

* Con var podemos re-asignar una variable pero es una mala práctica.
* Con let y const no podemos, aparecerá un error.
* Es una mala práctica crear una variable sin las palabras reservadas: var, let y const.  
  Si se asigna una variable dentro de una función sin las palabras reservadas será una variable global.
* La doble asignación de una variable también es una mala práctica.

# Local Scope

Lexical Scope / Ámbito Léxico: El intérprete de JavaScript funciona desde el ámbito de ejecución actual y funciona hasta encontrar la variable en cuestión. Si la variable no se encuentra en ningún ámbito, se genera una excepción.

Este tipo de búsqueda se llama ámbito léxico. El alcance de una variable se define por su ubicación dentro del código fuente, y las funciones anidadas tienen acceso a las variables declaradas en su alcance externo. No importa de dónde se llame una función, o incluso cómo se llama, su alcance léxico depende solo de dónde se declaró la función.

# Function Scope

Un pequeño resumen sobre cómo actúan las diferentes variables:

* Las variables escritas con la palabra clave **var** pueden ser redeclaradas, pero esto a futuro puede darnos problemas, así que es mejor no usarla.
* Las variables escritas con la palabra clave **let** no pueden ser redeclaradas, si lo haces mostrara un ***“error: esta variable ya ha sido declarada”*** , pero su valor puede ser reasignado:

**let** fruit = "apple";

fruit = "banana";

console.log(fruit); // banana

* Las variables escritas con la palabra clave **const** no pueden ser redeclaradas o reasignadas, ya que const quiere decir que su valor será constante, es decir que no va a cambiar.