**¿Qué es Hoisting y Scope?**

**¿Qué es?**

* **Scope**
  + Scope es un término dado al alcance de una variable ( Donde puedes acceder a una variable y donde no ), se dividen en dos tipos.
  + **Scope Global**
    - Determina como global a las variables que son declaradas fueras de bloques ( condicionales, ciclosm etc ) y Funciones.
    - **Características:**
      * Ya que el hoisting hace que todas las variables se coloquen al inicio de su bloque pertinente, en el caso de las variables definidas globalmente, permitirá que desde cualquier tipo de bloque y/o función puedas acceder a esta.  **No es recomendable usar este tipo de variables.**
  + **Scope Local**
    - Determina como local aquellas variables dentro de los bloques y/o funciones, las cuales solo se podrán acceder a estas desde dicho scope y/o bloque de código.
* **Hoisting**
  + Hoisting es un término dado al contexto de ejecución de JavaScript ( Especificamente refieriendose a creación y ejecución )
  + **¿Cómo funciona esto?**
    - Una manera de explicarlo es refiriéndose al término de que javascript al interpretar su código eleva las declaraciones al inicio de su **contexto**, esto incluye tipos de datos y funciones declaradas. Por esta razón puedes llamar funciones en javascript antes y después de ser declaradas.
    - ¿No lo entiendes? Entonces mira el siguiente ejemplo:
    - Var x = 10
    - ( () {
      * **Console.log(“x”, x);**
      * **Var x = 10;**
      * **Console.log(“x:”,x);**
    - });
    - A
    - En este ejemplo realmente la respuesta será la siguiente, la primera vez imprimirá undefined, y la segunda será 10, **¿Por qué?,** Es simple, Recuerdas que javascript sube las declaraciones al inicio de su contexto. Entonces javascript hará lo siguiente
    - Var x;
    - x = 5;
    - ( () {
      * **Var x;**
      * Console.log(“x”, x);
      * **x = 10;**
      * Console.log(“x:”,x);
    - });
    - **RECUERDA,**  Javascript solo elevará al inicio de su contexto las declaraciones de variables, no las inicializaciones.
    - Como puedes ver, primero dentro de scope del .ready javascript elevo el var x; más no la inicialización de x = 10; y no cogió el x = 5 puesto que no estaba dentor de su scope. Por ende en el primer console.log lanzará undefined