En javascript tenemos lo que se conoce como el “ámbito de las variables” o el “score” de las variables de cualquiera de las dos maneras estamos hablando de lo mismo, es decir, supongan que tenemos un dato en lo que llamamos el ámbito global bueno podrá ser accedido desde cualquier parte de nuestro código, si podemos acceder a él desde dentro de una función, desde dentro de algún otro algún, otro elemento de programación que no esté precisamente en el ámbito global. Pero al revés, en el ámbito local de una función, por ejemplo, un dato, una variable, no podrá ser accedido desde el ámbito global, dependiendo de la manera en que lo declaremos. Sí, la verdad es que este concepto, explicado así en abstracto, no es tan sencillo de entender como lo es cuando vamos al código.

¿Cómo creamos una función? Con la palabra reservada “function”. Ahora indicamos un nombre para la función, nuestra función se va a llamar “saludar”. Escribimos los paréntesis y abrimos y cerramos llaves, este es el cuerpo de la función, aquí entre las llaves, pues, nuestra función procesará la información de la manera en que nosotros le indiquemos, por lo pronto, simplemente vamos a decir que emita un mensaje, que haga un saludo.
Podríamos indicar aquí: “Hola Marcelo”. Bien, ahora podríamos mejor, más que “console log”, escribirlo en el documento, que aparezca aquí debajo perfecto ok ahora tenemos que invocar esta función, ¿cómo se invoca a una función?, es decir, ¿cómo se la pone en marcha? Simplemente llamándola por su nombre y los paréntesis. Y la función, al ser invocada, es ejecutada y corre todo el código que esté aquí adentro. Esta función es un tanto inflexible, no sirve para nada más que saludar a Marcelo, lo cual a Marcelo, tal vez le convenga, pero no es una función flexible, reitero, podríamos hacer que este parámetro que, mejor dicho, que ese dato, que el nombre de la persona a la que queremos saludar, ingrese a través de un parámetro, ¿si?, aquí podríamos, por ejemplo, decir que queremos saludar a, bueno, “name”, no importa el nombre que ingrese aquí.
Y, ¿por donde ingresan los parámetros a las funciones?, aquí, dentro de los paréntesis. Aquí podemos pasar uno o muchos parámetros, por ahora vamos a pasar solamente este. Okey. Llamamos a la función y nos dice “Hola, undefined”, esto es debido a que Javascript detecta que aquí hay una variable y cuando llamamos a la función, al no pasarle nada a esta variable, bueno, pues está vacía, no tiene un valor y recuerden que Javascript inicializa las variables con el valor de “undefined”, si es que nosotros no indicamos lo contrario.
Indicamos un valor de inicialización, por lo tanto, aquí esto tiene un par de soluciones. Lo primero es indicar un valor por defecto, para el caso de que no ingrese un parámetro, podríamos decir que el valor por defecto es “sin nombre”. Ahora, cuando llamamos a la función, que recibe un parámetro, al no enviar ese parámetro, toma el valor por defecto. Bien, pero vamos a pasarle efectivamente un parámetro que, para eso lo hemos definido. Vamos a pasar aquí un nombre y saludará a Lucrecia o Pedro, a María y así sucesivamente. Okey, muy bien, ahora esta función es una función un tanto impura, porque hace un procedimiento dentro de la función, pero en realidad lo que podríamos hacer para mejorarla, es que retornase un valor y luego nosotros podríamos manipular ese valor de muchas formas, porque tal vez quisiéramos enviarle este parámetro a la función, pero, quizás, no quisiéramos mostrar el dato por pantalla, sino almacenarlo en otro sitio, transformarlo. En fin, manipularlo de algún modo. Esta función es inflexible en el sentido de que, directamente, imprime algo por pantalla, pero podríamos hacer que en vez de imprimir algo por pantalla, nos retorne algo. Podríamos decir que retorne esto, ¿okey? La función al ser ejecutada, retornará lo que está a su derecha y ese valor, volverá al sitio donde se ha llamado la función, en este caso, nosotros estamos invocando la función aquí, la función corre, se ejecuta, retorna un mensaje

con la variable, pero esto no se está almacenando, no estamos atrapando ese valor de retorno. Podríamos hacer algunas cosas, por ejemplo, “const” “resultado, igual, saludar” y si ahora hiciéramos un “console log” de resultado, veríamos aquí en la consola, “María”, ¿okey?. Bien, para que viéramos todo esto tendríamos que ponerlo, por ejemplo, así entre paréntesis, ¿si? Okey, bien, no estamos almacenando el resultado, pero podríamos hacer alguna otra cosa, podríamos, vamos a poner un más, en realidad más, ¿okey? Estamos aquí, concatenando texto, ¿okey? Concatenamos la cadena “Hola” con la cadena que está almacenada dentro de la variable “name” que es, en este caso, “María”, pero podría ser otro nombre, cualquiera, no importa. Bueno, muy bien, ahí estamos entonces, atrapando el valor de retorno y luego podemos mostrarlo o podemos hacer alguna otra cosa con él. Aquí la función es un tanto más pura que antes, hay otra manera de definir funciones y es con “arrow functions” o funciones de flecha. Definimos un nombre, por ejemplo, “despedir”, ponemos igual los paréntesis y aquí adentro vamos a indicar si es que hay algún parámetro, bueno, vamos a poner otra vez “name” como parámetro, dibujamos una flecha, por eso se llaman funciones de flecha o “arrow functions” y luego es igual que con la palabra reservada “function”, lo que hagamos aquí será lo que se ejecute. Podríamos retornar, no quiero borrarlo, esto quiero copiarlo, por ejemplo, “Hasta luego” y aquí, cuando ejecutemos “Saludar”, el resultado se va a almacenar aquí, el retorno se va a almacenar en resultado, podríamos decir “Intro” y luego podríamos crear una constante “outro” igual, “despedir” y también podemos despedir a María, o a quien sea, no importa, esto es simplemente un ejemplo. Y luego aquí, podríamos indicar, por ejemplo, “mostrar por pantalla” “intro” y a “outro” en la misma línea o podríamos hacer dos “console log”, no importa, esto es lo de menos y aquí ustedes pueden ver cómo invocamos a una función el resultado. El retorno es almacenado en esta variable y luego, esta variable es mostrada por pantalla. Para “outro”, la hemos diseñado como una “arrow function”, pero funciona exactamente igual, como ustedes pueden ver y hacemos lo mismo, la llamamos con un parámetro. Aquí le pasamos este argumento y luego es impresa también por pantalla, junto con la otra variable.

Muy bien, ahí tenemos entonces las formas de definir variables y de llamar a las variables. Ahora, vamos a notar lo siguiente. Supongamos que tenemos una función que se llame de esta manera, no importa el nombre, es lo de menos y aquí queremos, puedes hacer alguna operación, por ejemplo, quisiéramos mostrar por pantalla, okey, “A más B”. Bien, perfecto, esto no va a mostrar nada por pantalla, primero porque no estamos llamando la variedad, la función aquí la podemos llamar, esto no va a hacer nada, porque A y B no están definidas, perfecto, entonces vamos a definir A y B: “let A igual 10”, “let”, podríamos hacerlo con “const”, es indistinto, “B igual 20” y aquí me está dando el resultado de 30. A pesar de que estas variables están fuera de la función, se encuentran en el ámbito global, todo lo que está en el ámbito global puede ser leído dentro de una función, ¿okey?. Esto funciona, no es del todo correcto, pero funciona. Esto por supuesto que también va a funcionar, aquí tenemos las variables dentro del ámbito de la función y es más correcto, porque son variables que usa la función aquí adentro, por lo tanto, es más correcto que estén aquí adentro.

Bien, esto es lo que entendemos como “scoped” o alcance o ámbito. También el ámbito de las funciones, lo que está entre llaves, sin embargo, aquí dentro puedo leer valores que estén afuera, en el ámbito global, “const a igual 10” y aquí podríamos borrar “A” y esto va a seguir funcionando, porque está leyendo “A” desde el ámbito global y “B” la está leyendo desde el ámbito interior.

¿Qué ocurre si tenemos aquí otra variable “A”, con otro valor distinto? 20, ¿qué va a leer? Bien, por más que pueda acceder al ámbito global, la función siempre, pongamos que

prefiere variables de su propio ámbito, por lo tanto, si tenemos una variable “A” en este ámbito y una variable “A” en el ámbito exterior o global, la función va a leer siempre primero la del ámbito interior, ¿okey? Bien.
Ahora, ya que estamos hablando de “scoped” o de ámbito, veamos lo siguiente:
¿Qué ocurre si, fuera de la función, queremos imprimir el valor de “A” o de “B”?, por ejemplo, esto nos va a dar un un error de referencia, porque nos dice que “A” no está definida y ustedes podrían preguntarse, ¿cómo que no está definida, si aquí está definida?. Okey, analicemos esto, este par de llaves indican un ámbito, lo que está aquí está en este ámbito, este es el ámbito de la función. Todo lo que está afuera, no puede acceder de manera directa a lo que está aquí adentro, pero si al revés. Por eso recuerden, que si tenemos ahí “B” o uno de los dos fuera de este ámbito, por ejemplo, esto vamos a llevarlo fuera, podemos ponerlo arriba o abajo, en este caso sería indistinto, nuestra función va a correr de todas maneras. Ahora no corre, porque nuestro programa llega a esta línea y se rompe. Pero si bajamos esto, va a llegar a ejecutar la función, 20 más “B”, que está fuera, igual a 40 y luego se va a romper, ¿por qué se rompe?, porque no puedo, desde el ámbito global, acceder al ámbito de la función. Desde afuera no puedo acceder adentro, pero si al revés, desde dentro de la función puedo leer lo que está afuera en el ámbito global.