En javascript tenemos lo que se conoce como el "ámbito de las variables" o el "scope" de las variables de cualquiera de las dos maneras estamos hablando de lo mismo, es decir, suponga 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 está aquí adentro. Esta función es un tanto inflexible, no sirve para nada más que saludar a Marcelo, la 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, ¿sí?, aquí podríamos, por ejemplo, decir que queremos saludar a, bueno, "name", no importa el nombre que ingrese aquí.
Y, ¿por dónde 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. La primera 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