**Variable**

Una variable es almacenar un valor o dato para ser utilizado posteriormente. La variable se puede modificar o configurar y adaptar a cualquier tipo, no es necesario declarar su tipo al iniciar su valor. En JavaScript la palabra clave es var, seguida de un espacio y el nombre de la variable.

Podemos agregar, no se pueden utilizar palabras que están establecidas por el lenguaje de programación como nombres de variables.

Ejemplo:

* var numero = 3;
* var nombre = ‘Juan’;
* const deporteFavorito = ‘Soccer’;

**String**

Los String son bloques de texto o cadena de texto, siempre se definirán entre comillas (dobles o simples), donde cualquier texto entre comillas es una cadena o String.

**var** nombreAmigo = ‘Martínez’;

**Funciones**

Las funciones son una parte muy importante de todos los lenguajes de programación y sobre todo en JavaScript. Llamados objetos invocables, por lo que tienen las mismas propiedades que cualquier objeto.

Al contar con un conjunto de variables, necesitamos funciones para calcular, cambiarlas o hacer algo con ellas.

Para construir una función lo manejaremos de la siguiente manera:

* function nombreFuncion() {}

Una función comenzara con la palabra clave function, esto le dice a lo que sea que esté ejecutando tu programa que lo que sigue es una función y que debe tratarse como tal. Después de eso viene el nombre de la función, nos gusta dar nombres de funciones que describen lo que hacen. Luego viene un paréntesis abierto y uno cerrado. Y finalmente, abra y cierra las llaves. Entre estos corchetes es donde irá todo nuestro código a ejecutar.

Ejemplo:

Function logHola() {

console.log(‘hola’);

}

logHola();

En este ejemplo declaramos una función logHola y la configuramos en console .log(‘hola’).

Entonces podemos ver qué para ejecutar esta función, necesitamos escribir el nombre y los paréntesis. Esta es la sintaxis para ejecutar una función. Una función siempre necesita paréntesis para ejecutarse.

**Argumentos**

Ya identificamos como se ejecuta una función básica, comenzamos a pasarle argumentos.

function logHola(*nombre*){

console.log(‘Hola,’ + nombre);

}

logHola(*‘Martin’*);

Si agregamos una variable a los paréntesis cuando declaramos la función, podemos usar esta variable dentro de la función. Iniciamos el valor de esta variable pasándola a la función cuando la llamamos. Entonces en este caso nombre = ‘Martín’. También podemos pasar otras variables a esto:

Function logHola(*nombre*){

console.log(‘Hola, ${*nombre*}’);

}

var miNombre = ‘Antonio’;

logHola(*minombre*);

Podemos agregar múltiples argumentos agregando una coma entre ellos:

function sumaDosNumeros(*a, b*) {

var suma = a + b;

return suma;

}

sumaDosNumeros(*1, 5*);

**Return**

No vamos a utilizar console.log con todo lo que salga de una función. Lo más probable es que queramos devolver algo. En el caso anterior es la suma de los dos números. Piensen en la declaración de retorno (“return”) con la única forma en que los datos escapan de una función. No se puede acceder a nada más que a lo que devuelve fuera de la función. Cuando una función golpea una declaración de retorno, la función detiene inmediatamente lo que está haciendo y devuelve lo especifica.

Ejemplo:

function dividirDosNumeros(a, b) {

var producto = a / b;

*return producto;*

}

dividirDosNumeros(6,3);

console.log(producto);

**Declaraciones ‘if’**

Maneja un flujo de control, es decir, se hace una comparación y si esta comparación mediante operadores de comparación (>, <, <=, >=, ===, ¡==) es true ejecuta el código que se contiene entre llaves, sino es true sigue el proceso después del bucle if.

function puedeManejar (edad) {

If (edad>17)

{

return true;

}

return false;

}

puedeManejar(22);

**Boolean**

Los booleanos proviene de la lógica de Boole. Es un concepto que alimenta el código binario y el núcleo de computadora. Significa que tiene dos opciones, activar o desactivar, 0 o 1, verdadero o falso. En JavaScript usamos booleanos para significar verdadero o falso. Esto puede parecer simple al principio, pero puede complicarse más adelante.