## VARIABLES

|  |  |  |
| --- | --- | --- |
| Sintaxis | Uso | Ejemplo |
| var | Le indica a JavaScript que vamos a declarar una variable de tipo var a la cual le podemos asignar un valor. | var nombre = ‘Hackerman’; |
| let | Solo será accesible en el bloque de código en el que fue declarada. | let contador = 0; |
| const | Al igual que let solo es accesible en el bloque de código en el que fue declarada. Además, no  podemos cambiar su valor. | const url =http://digitalhouse.com.ar’; |

## TIPOS DE DATOS

|  |  |  |
| --- | --- | --- |
| Tipos | Explicación | Ejemplo |
| cadenas de caracteres  (string) | Cadenas de textos. Se escriben entre comillas dobles o simples. | let nombre = ‘Juan’ |
| lógicos o booleanos | Su valor puede ser true o false (verdadero o falso). | let esArgentino: true; |
| objeto (object) | Son colecciones de datos y en su interior pueden existir los anteriores. | let perro = {nombre: ‘Firu’,  patas: 4} |
| array | También son colecciones de datos y se escriben entre corchetes []. | Let pelis = [‘Thor’, ’La Ouija’, ‘etc’] |
| NaN (Not a number) | No es un número. | return 2 \* ‘Hola’ = NaN |
| NULL (Nulo) | Los asignamos nosotros para indicar un valor vacío o desconocido. | Let pasajeros = null; |
| UNDEFINED (Valor sin  definir) | Las variables tienen un valor indefinido hasta que les asignamos un valor. | Let comidaHoy; |

## JSON

|  |  |  |
| --- | --- | --- |
| Tipo | Explicación | Ejemplo |
| JSON.parse() | Recibe un formato JSON y lo transforma en Javascript |  |
| JSON.stringify() | Recibr un dato Javascript y devuelve una cadena de texto en formato JSON |  |

## MÉTODOS DE STRINGS

|  |  |  |
| --- | --- | --- |
| Tipo | Explicación | Ejemplo |
| .slice() | Toma una porción de una cadena de texto. Se le pasan 2 parametros numéricos (el 1ro donde empieza la porción y el segundo en donde termina) | let clima = “Lindo día”;  console.log(climaslice(0,4);  “Lindo” |
| .indexOf() | Busca un string que le indicamos y nos muestra la posición de lo buscado. Si no existe devuelve -1 |  |
| .trim() | Elimina espacios en blanco antes y después del primer caracter. No recibe parámetros | “ Soy un humano ”.trim()  “Soy un humano” |
| .split() | Convierte un string en un array. Recibe como parámetro el divisor de cada elemento. | “Soy del planeta tierra”.split(“ “);  [“Soy”, “del”, “planeta”, “tierra”] |
| replace | Permite remplazar una porción de texto dentro de un string. Recibe 2 parámetros (1ro la porción de texto que buscas, 2do por lo que la querés remplazar) | “Aguante Phyton” .replace(“Phyton”,”Javascipt”)  “Aguante Javascript” |

## OBJETOS LITERALES

|  |  |  |
| --- | --- | --- |
| Tipo | Explicación | Ejemplo |
| Método | Si un objeto literal contiene una función, decimos que es un método. Para llamarlo utilizamos objeto.metodo() | Let tenista = { nombre: ‘Roger’  Saludar: function(){  Return ‘Hola, me llamo Roger!’  } |
| Funciones constructoras | Permite armar un molde para crear todos los métodos que necesitemos. Las escribimos con mayúsculas para identificarlas.  Para “usar” este molde hacemos: | Function Auto (marca,modelo){  This.marca = marca;  This.modelo = modelo;}  Let miAuto = new Auto (‘Ford’,’Falcon’); |

## ARROW FUNCTIONS

|  |  |  |
| --- | --- | --- |
|  | Explicación | Ejemplo |
| Qué son? | Nos permiten escribir funciones de una forma más compacta. Estas son anónimas, así que para nombrarlas deben ser valor de una variable. Si recibe 1 solo parámetro, no necesita paréntesis. | Function sumar (a,b) {return a + b}  ***pasa a ser***  Let sumar = (n1,n2) => n1 + n2 |

## IF TERNARIO

|  |  |  |
| --- | --- | --- |
|  | Explicación | Ejemplo |
| Qué es? | Similar al if, no necesita el uso de las palabras if, else o {}. Si es V ejecuta la primer expresión, si es F ejecuta la segunda. | Condición ? expresión1 : expresión2 ; |

## SWITCH

|  |  |  |
| --- | --- | --- |
|  | Explicación | Ejemplo |
| Qué es? | Evalúa muchas posibilidades de una forma más compacta. Está compuesto por una expresión, y seguido de los diferentes casos a evaluar. Para separar cada caso utilizamos el break.  En caso de que ninguno sea correcto, se utiliza el default. | Switch (expresión) {  case valorA:  break;  case valorB:  break;} |

MÉTODOS DE ARRAYS

|  |  |  |
| --- | --- | --- |
| Tipo | Explicación | Ejemplo |
| Map() | Recibe un callback de parámetro, recorre el array y devuelve uno nuevo modificado. Las modificaciones son las que programemos en el callback. | Let números = [2,4,6]  Let elDoble = números.map(function(num){  Return num \* 2;)} |
| Filter() | Recibe un callback, recorre el array y los filtra de acuerdo a la condición establecida. Devuelve un nuevo array únicamente con los elementos que cumplan la condición. | Let edades = [22,8,17,14,20]  Let mayores = edades.filter(function(edad){  Return edad > 18 }) |
| Reduce() | Recorre el array y devuelve un único valor. Recibe un callback que tiene 2 parámetros: un acumulador y un elemento actual que esté recorriendo. | Let nums = [5,7,16]  Let suma = nums.reduce(function(acum,num){  Return acum + num}) = 28 (se acumula el res) |
| ForEach | Itera sobre un array, recibe un callback, pero no retorna nada. Nosotros le indicamos a la función dentro del callback el comportamiento que queremos que tenga el forEach. | Let países = [‘Argentina’,’Cuba’,’Perú’];  Países.forEach(function(país){  Console.log(país);  }) |