JAVASCRIPT

Variables

Const = Variable constante. No se le puede **reasignar** un valor (**final** en Java).
Let = Variable reasignable.

**typeof = saca el tipo de variable.*

Diferencias entre function y method

Ambos son **bloques de código** a los que puedes llamar pero un **método** pertenece a un **objeto** concreto de una clase, a diferencia de la **función** que **NO** está ligado a un objeto y se puede llamar sin referenciar una clase.

Se podría decir que lo más parecido a **function** es un **método static**.

**Math.floor() redondea hacia abajo.*

Diferencias entre parámetro y argumento

Parámetro = Órden, tipo de dato que se espera recibir

Argumento = Dato explícito y directo que se recibe

Function expression

¿En qué consiste una expresión de función?

Guardamos la función en una constante, que recibirá el nombre de dicha función, perdiendo su nombre y haciendo pasar su tipo a “función anónima”.

Arrow function

Simplifica, ahorra código y permite los returns explícitos

```
const sumar = (a, b) => {
      return a + b;
}
```

Con return explícito:

```
const sumar = (a, b) => a + b
```

Recursividad

Consiste en funciones que se llaman a **ellas mismas** para ejecutarse. Parten de una **condición base** (*normalmente 0 o 1*) en la que, si se cumple, se termina de ejecutar el bloque. De **no** darse esta condición base, **se cumple** con el código desarrollado.

```
function factorial (n) {
      if (n === 0 || n === 1) {
      return 1
      } else {
       return n * factorial(n - 1)
   }
}
```

Arrays en JavaScript

¿Cómo se declara un array en JS?

Java: `int[] numeros = new int[longitud];`

JavaScript: `const numeros = [longitud];`

Peculiaridad: No hace falta hacer un bucle para mostrar los elementos de un array. Directamente con console.log().

Métodos de arrays

.length = Devuelve longitud del array

**.length es reasignable (borra elemento del final)*

.push = Añade un nuevo o varios elemento/s al final del array

.pop = Elimina y guarda el último elemento del array

.shift = Lo mismo que el .pop pero con el primer elemento

.unshift = Añade uno o varios elementos al principio

.concat = Concatena valores de arrays.

```
      const numeros1 = [1, 2, 3]
      const numeros2 = [4, 5]

      const numCompletos = numeros1.concat(numeros2)
```

Operador spread (...)

Concatena los valores del array

```
const numCompletos = [...numeros1,...numeros2]
```

Estructuras de control de arrays

For…of

Se declara una variable que iterará sobre el rango del array, perdiendo el control sobre el iterador.

```
for (const fruta of frutas)
```

forEach

Bucle que permite ejecutar una función, entendiéndose de esta manera que, por cada iteración del bucle, realizará una función. La semántica es:

```
frutas.forEach(function (elemento, index, arrayOriginal)
```

Ejemplo:

```
frutas.forEach(function (el, index) {
      console.log('index: ' + index) //ÍNDICE
      console.log('elemento: ' + el) //ELEMENTO
      }
```

Más métodos y funciones

.indexOf = En una variable devuelve el índice (posición) de un elemento que recibe como parámetro. Si dicho elemento no existe, devuelve un -1.

```
const posicionCorazon = emojis.indexOf('😍')

console.log(posicionCorazon) // -> 2
```

.includes = booleano que devuelve en función de si encuentra un valor o no.

```
const tieneCorazon = emojis.includes('😍')
```

.some = devuelve true o false en función de si al menos uno de los elementos de un array cumple con una condición.

```
const names = ['Leo', 'Isa', 'Ían', 'Lea']

const tieneNombreLargo = names.some(name => name.length > 3)
console.log(tieneNombreLargo) // -> false
```

.every = Retorna true si todos los elementos cumplen con una condición

```
const numbers = [2, 4, 7, 10, 12]
const todosSonPares = numbers.every(number => number % 2 === 0)
console.log(todosSonPares) // -> false
```

.find = Busca el primer elemento que cumpla con una condición

.findIndex = Igual que el find pero en lugar de devolver el elemento, devuelve su índice

Ejercicio

```
function acabanEnA(words) {
  const palabras = words.every(word => word.endsWith('a'))
  return palabras
}
```

Ordenamiento de arrays

.sort = Ordenamiento según valor **alfabético** (transforma todos los valores en cadena de texto). **Problema:** Ordenamiento inesperado de valores numéricos. **Solución:** Uso de funciones:

```
numeros.sort((a, b) => a - b)
```

Se trata de una función incluida dentro del método sort de JavaScript. Este método ordena numéricamente los valores de un array.

.toSorted = Ordenamiento de array a través de una copia del array original.

Transformación de arrays

.filter = Recibe una función como parámetro y devuelve los valores que cumplan con la condición que ejecuta la función. **Casos de ejemplo**: Filtrar y crear un array con los números pares; crear un array con las palabras que contengan la letra a

```
const strings = ['hola', 'adiós', 'casa', 'coche', 'perro', 'gato']

// en las cadenas de texto podemos usar el método `includes`
// para saber si una cadena contiene otra:
const stringsWithA = strings.filter(string => string.includes('a'))

console.log(stringsWithA) // ['hola', 'adiós', 'casa', 'gato']
```

.map = Recibe función que, dándose la condición, devuelve un array con cada elemento transformado a placer. Ejemplo: Dándose la condición, que devuelva los números multiplicados.

.map + .filter = Caso de uso: Doblar el valor de los números y quedarnos con los mayores a 5.

.reduce =

Ejercicio

Sacar las palabras con una longitud mayor al valor del índice de una palabra dada dentro de una lista de palabras:

```
function buscaPalabras(words, word) {
  let long = words.indexOf(word)
  let palabrasLargas = words.filter(words => words.length > long)
  return palabrasLargas
}
```