Guia de 40 ejercicios JavaScript

1 - Retornar por consola el string “Hola Mundo”.

2 - Pedir por prompt un número y guardarlo en una variable, luego pedir un segundo número y guardarlo en otra variable. Crear una tercera variable que sume los valores pedidos y retornarlos por consola.

3 - Pedir por prompt un string y guardarlo en una variable, a continuación pedir un segundo string y guardarlo en otra variable. Luego crear una variable que compare ambos strings y retorne “true” en caso de que sean iguales o “false” en caso de que no lo sean. Agregar una variable más que compare la longitud de ambos strings y retorne “true” en caso de que ambos tengan la misma longitud, o “false” en caso contrario.

4 - Pedir por prompt una número y luego crear una función que nos diga si dicho número es divisible por 5 o no. Mostrar el resultado en la consola.

5 - Pedir por prompt una temperatura en grados Farenheit y luego crear una función que convierta la dicha temperatura a grados Celsius. Finalmente mostrar por consola el resultado de la conversión.

6 - Crear un programa donde se introduzcan los tres ángulos internos de un triángulo y se determine si el triángulo es válido o no.

7 - Crear un programa que le pida al usuario dos números en un Prompt y luego muestre en por consola cuál es el número mayor.

8 - Crear un programa que determine si un año dado es bisiesto. En caso de que lo sea retornar por consola “El año ingresado es bisiesto”, y en caso de no serlo retornar “El año ingresado no es bisiesto”.

9 - Crear un programa que solicite una edad y dependiendo del valor ingresado retorne por consola lo siguiente:
- “Es un niño” si la edad es menor a 13.
- “Es un adolescente” si la edad ingresada está entre 13 y 17.
- “Es un adulto” si la edad está entre 18 y 110.
- “No es una edad válida” en el caso de que no se cumpla ninguna de las condiciones anteriores.

10 - Crear un programa que pida un color al usuario y retorne por consola el string correspondiente:
- En caso que el color recibido sea “azul”, → Devuelve “blue”
- En caso que el color recibido sea “rojo”, → Devuelve “red”

- En caso que el color recibido sea “verde”, → Devuelve “green”
- En caso que el color recibido sea “naranja”, → Devuelve “orange”
- En caso que el color recibido sea “amarillo”, → Devuelve “yellow”
- En caso que el color recibido no sea ninguno de los anteriores → Devuelve “Color not found”

Se debe usar el statement Switch.

11 - Crear una función llamada “operadoresLogicos” con 3 números por parámetros que retorne lo siguiente:

- Si num1 es mayor a num2 y a num3 y además es positivo, retornar ---> "Número 1 es mayor y positivo"
- Si alguno de los tres números es negativo, retornar ---> "Hay negativos"
- Si num3 es más grande que num1 y num2, aumentar su valor en 1 y retornar el nuevo valor.
- 0 no es ni positivo ni negativo. Si alguno de los argumentos es 0, retornar "Error".
- Si no se cumplen ninguna de las condiciones anteriores, retornar false.

12 - Crear una función que reciba un número y retorne “true” si dicho número es primo. Caso contrario devuelve “false”.
Pista: un número primo solo es divisible por sí mismo y por 1
Pista 2: Puedes resolverlo usando un bucle `for`
Nota: Los números 0 y 1 NO son considerados números primos

13 - Escribí un loop que imprima en la consola los números del 1 al 100.
Deberá cumplir las siguientes condiciones: si el número a imprimir es múltiplo de 3, debe mostrar en la consola el string 'Fizz'. En cambio, si es múltiplo de 5, debe mostrar:
'Buzz'. Por último, si es múltiplo de ambos debe mostrar: 'FizzBuzz’

Ejemplo de output:

1
2
Fizz
4
Buzz
Fizz
...
14
FizzBuzz
16
*/

14 - Creá una función que tenga la misma funcionalidad que el FizzBuzz anterior, pero que reciba por parámetro las palabras a imprimir (en vez de Fizz y Buzz) y los números con los que se activan y el número máximo de iteraciones.

Input: fizzBuzz('Plata', 'Forma', 2, 7, 17)

Output esperado:

1
Plata
3
Plata
5
Plata
Forma
...
PlataForma
15
Plata
17
*/

15 - Crear una función llamada “divisibles” que reciba dos parámetros:

- Un arreglo de números
- y un divisor.

Esta deberá retornar un arreglo con los números que sean divisibles por el segundo parámetro.

Ejemplo:
console.log(divisibles([1, 2, 3, 4, 5, 6], 2)); //debe retornar [2, 4, 6]
console.log(divisibles([1, 2, 3, 4, 5, 6], 3)); //debe retornar [3, 6]

16 - Crear una función llamada “rango” que reciba tres parámetros:

- Un número de comienzo
- Un número de final
- y un sumador.

Esta deberá retornar un arreglo con los números que estén entre el de comienzo y el de final, sumando de a tanto como sea el sumador.

Ejemplo:
rango(1,10,3) => [1, 4, 7, 10]

17 - Crear una función llamada “nuevoArreglo” que reciba un número como parámetro y que devuelva un nuevo arreglo con tantos elementos como el número que le hayas pasado.

Ejemplo:
nuevoArreglo(5) debe retornar [1,2,3,4,5]
nuevoArreglo(10) debe retornar [1,2,3,4,5,6,7,8,9,10]

18 - Crear una funcion llamada “breakStatement” que reciba un número e itere en un bucle aumentando en 2 hasta un límite de 10 veces.
Guardar cada nuevo valor en un array y devolverlo.
Si en algún momento el valor de la suma y la cantidad de iteraciones coinciden, debe interrumpirse la ejecución y devolver: "Se interrumpió la ejecución"
Pista: usá el statement 'break'

19 - Crear una función llamada toTime() que tome como argumento un número entero (segundos).
La función debe convertir el valor recibido en un string ("") que describa cuantas horas y minutos comprenden esa cantidad de segundos
El resultado debe tener el siguiente formato: "X hora (s) y X minuto (s)"

Ejemplos:

toTime(3600) ==> "1 hour(s) and 0 minutes(s)"
toTime(3500) ==> "0 hour(s) and 58 minutes(s)"
toTime(323500) ==> "89 hour(s) and 51 minutes(s)"

Pista: primero llevar segundo a horas y después llevar segundos a minutos ; tenemos que hacer las dos operaciones

20 - Sumatoria
Debés crear una función llamada `sumattion` que reciba un número como parámetro y que devuelva la sumatoria de todos sus números anteriores, incluso ese mismo.

Ejemplo:
sumattion(3) debe retornar 6 porque hace (1 +2 +3)
sumattion(8) debe retornar 36

21 - Crea una función llamada "abbrevName" que reciba como parámetro un string. El string recibido siempre tiene que incluir un espacio. La función debería convertir el string recibido en iniciales.

Ejemplos:
abbrevName("Sam Harris") ==> "S.H"
abbrevName("Evan Cole") ==> "E.C"
abbrevName("David Mendieta") ==> "D.M"

Pista: El método toUpperCase() devuelve el valor convertido en mayúsculas de la cadena que realiza la llamada.
El método split() divide un objeto de tipo String en un array (vector) de cadenas mediante la separación de la cadena en subcadenas.

22 - Crear una función llamada “moveZeros” que reciba un arreglo como parámetro y devuelva otro con los números “0” ordenados al final.

Ejemplo:
moveZeros([false,1,0,1,2,0,1,3,"a"]) debe retornar [false,1,1,2,1,3,"a",0,0]
moveZeros([1,2,0,1,0,1,0,3,0,1]) debe retornar [1,2,1,1,3,1,0,0,0,0]

23 - Crear una función llamada “arregloDeObjetos” que reciba un número como parámetro y devuelva un arreglo de objetos que tengan una propiedad llamada “valor” que contenga el valor del número y sus anteriores
.
Ejemplo:
- arregloDeObjetos(5) debe retornar [{valor: 1}, {valor: 2}, {valor: 3}, {valor: 4}, {valor: 5}]
- arregloDeObjetos(3) debe retornar [{valor: 1}, {valor: 2}, {valor: 3}]

24 - Crear una función pluck que tome dos parámetros, un arreglo de objetos y el nombre de una propiedad. La función devolverá un nuevo arreglo solo con los valores dentro de la propiedad recibida.

Ejemplo:
var productos = [{ name: 'TV LCD', price: 100}, { name: 'Computadora', price: 500 }]

pluck(productos, 'name') // ['TV LCD', 'Computadora']
pluck(productos, ‘price’) // [100, 500]

25 - Crear la función “sumArray” que tome dos parámetros, un arreglo de números ordenados y un número.
La función devolverá true, si cualquier combinación de dos números dentro del arreglo suman el número del segundo parámetro. Sino, devolverá false.

Output esperados:

console.log(sumArray([2, 5, 7, 10, 11, 15, 20], 13)); // true 2+11 suman 13
console.log(sumArray([2, 5, 7, 10, 11, 15, 20], 14)); // false

26 - Sacar el valor total de todos los artículos que se encuentren en la tienda, pero que estén separados por sección, cada artículo diferenciado de otro sin estar en el mismo arreglo.
Crear una nueva función llamada “totalDeArticulos” que como parametro que tenga, haga referencia a un arreglo de productos de nuestro ecommerce.
La función en general tendrá que devolver un nuevo arreglo con objetos que tengan el nombre de cada producto y el valor total de todos los artículos que se encuentren en la tienda.

Utilizar la siguiente variable:

var ecommerce = [{ nombre: "Samsung TV", precio: 6000, articulos:10}, { nombre: "DELL notbook", precio: 4000, articulos:30 }, {nombre:"Auriculares Sony", precio: 1500, articulos:15}, {nombre:"Monitor Philips", precio:12000, articulos:20}, {nombre:"Teclado logitech", precio: 3000, articulos:5}]

Output esperado:

TotalDeArticulos(ecommerce) // Debe retornar (5) [{…}, {…}, {…}, {…}, {…}]
{Samsung TV: 60000}
{DELL notbook: 120000}
{Auriculares Sony: 22500}
{Monitor Philips: 240000}
{Teclado logitech: 15000}

27 - Crear una función llamada “gananciaTotal” que recibe como argumento un array con los balances de un periodo y tal como lo indica su nombre retorna la ganancia total de ese periodo.

Utilizar la siguiente variable:
```
let balancesUltimoSemestre = [
  { mes: "julio", ganancia: 50 },
  { mes: "agosto", ganancia: -12 },
  { mes: "septiembre", ganancia: 1000 },
  { mes: "octubre", ganancia: 300 },
  { mes:  "noviembre", ganancia: 200 },
  { mes: "diciembre", ganancia: 0 }
];
```

Output esperado:
1538

28 - Crear una función que retorne la cantidad de balances positivos utilizando la variable “balancesUltimoSemestre” del ejercicio anterior.

Output esperado: 4

29 - Crear una función llamada “contarHasta” que pide un número positivo por consola y retorna una lista con los números desde el 0 hasta el número recibido. Utilizar el bucle While para hacerlo.

30 - Crear un programa que pida al usuario que teclee un número y se repita hasta que este teclee un 0. Además, al finalizar mostrará por consola la suma de todos los números tecleados.

31 - Crear una nueva versión de la función “esPrimo” utilizando el bucle While la cual va a determinar si el número que ingresa el usuario es primo o no. Recordar que un número primo es el que solo puede dividirse por sí mismo y por la unidad.
Pista: Puedes utilizar break para detener la ejecución de las iteraciones cuando sea necesario.

32 - Crear un programa que solicite por consola al usuario que ingrese la letra “F” (en mayúscula) y en caso de que este ingrese un valor distinto se lo vuelva a pedir hasta que ingrese la letra correcta. Una vez ingresada la letra F devolver por consola el siguiente mensaje: “Lo lograste!”
Pista: para este ejercicio deberás utilizar el bucle Do While

33 - Crear un programa que solicite por consola al usuario que ingrese primero un número y luego otro. Este debe retornar el valor de la suma y preguntar al usuario si quiere repetir la operación ingresando “SI” o “NO”.
Si el usuario ingresa “SI” el programa comienza nuevamente.
Si el usuario ingresa “NO” el programa finaliza.
En caso de que el usuario ingrese una cadena diferente, debe volver a realizar la pregunta hasta que el input coincida con uno de los strings “SI” o “NO”
Pista: para este ejercicio necesitarás utilizar bucles Do While. Y se puede usar utilizar un bucle dentro de otro.

34 - Crear una función llamada “doubleFilter” que reciba como parámetro un arreglo de objetos, un continente, y un número de población. La función filtra el arreglo solo con los países que sean del continente pasado por parámetro, y además, los que su población sea mayor o igual a la del parámetro.o

Output esperado:

```
doubleFilter(paises, "europa", 4000001);
[
  {nombre: "españa", continente: "europa", poblacion: 29000000},
  {nombre: "alemania", continente: "europa", poblacion: 33000000},
```

```
   {nombre: "francia", continente: "europa", poblacion: 65000000},
   {nombre: "grecia", continente: "europa", poblacion: 12000000},
]
```

Utilizar el siguiente arreglo de objetos:

```
var paises = [
  {nombre: "argentina", continente: "sudamerica",  poblacion: 40000000},
  {nombre: "brasil", continente: "sudamerica", poblacion: 300000000},
  {nombre: "venezuela", continente: "sudamerica", poblacion: 25000000},
  {nombre: "chile", continente: "sudamerica", poblacion: 10000000},
  {nombre: "australia", continente: "oceania", poblacion: 18000000},
  {nombre: "nueva zelanda", continente: "oceania", poblacion: 8000000},
  {nombre: "china", continente: "asia", poblacion: 1000000000},
  {nombre: "tailandia", continente: "asia", poblacion: 32000000},
  {nombre: "vietnam", continente: "asia", poblacion: 23000000},
  {nombre: "españa", continente: "europa", poblacion: 29000000},
  {nombre: "alemania", continente: "europa", poblacion: 33000000},
  {nombre: "francia", continente: "europa", poblacion: 65000000},
  {nombre: "portugal", continente: "europa", poblacion: 4000000},
  {nombre: "grecia", continente: "europa", poblacion: 12000000},
];
```

35 - Ahora deben refactorizar la función doubleFilter, que recibirá los mismos tres parámetros pero en vez de devolver un arreglo con los nombres de los países que cumplan con las condiciones, devolverá un objeto con una key ‘nombres’ que tendrá como valor un arreglo con los nombres de los países y otra llamada ‘población total’ cuyo valor sea la suma de las poblaciones de los países filtrados.

Ejemplo:

doubleFilter(paises, ‘sudamerica’, 30000000) debe retornar

```
{
  nombres: [‘argentina’, ‘brasil],
  poblacion total: 340000000
}
```

36 - Crear un programa de apuestas con dados el cual le va a pedir al usuario un número al que quiera apostar y la cantidad que quiere apostar.
Con ayuda de la función “Math.random()” se debe generar un valor del 1 al 6 que represente el valor obtenido al tirar el dado y se asigne a la variable “dado”.
El usuario empieza con 50 pesos para apostar.
Si acierta el número gana el doble de lo apostado. Caso contrario pierde lo apostado.
Además debe cumplir las siguiente condiciones:

- ☐ Al inicio del programa se debe mostrar por pantalla cuánto dinero tiene el usuario para apostar.
- ☐ En caso de que el usuario no ingrese ningún número por el cual apostar se toma como valor por defecto “1”.
- ☐ En caso de que el usuario no ingrese ningún monto para la apuesta se toma como valor por defecto “20”
- ☐ Se debe mostrar por pantalla el número que ha salido.
- ☐ En caso de que haya acertado la apuesta mostrar por pantalla cuánto dinero ha ganado.
- ☐ En caso de que no haya acertado la apuesta mostrar por pantalla cuánto dinero ha perdido y cuánto dinero le queda.
- ☐ Si la persona se queda sin dinero mostrar por pantalla que ha perdido el juego y terminar la ejecución del programa.
- ☐ Si la persona llega a 200 o más mostrar por pantalla que ha ganado el juego y terminar la ejecución del programa.

37 - Crear una función que reciba un string por parámetro y retorne si esa cadena está formada sólo por mayúsculas, sólo por minúsculas o por mayúsculas y minúsculas.

38 - Crear una función que determine si una cadena de texto que se pasa como parámetro es un palíndromo, es decir, si se lee de la misma forma desde la izquierda y desde la derecha.
Ejemplo de palíndromo complejo: "La ruta nos aporto otro paso natural".
Output esperados:

console.log(esPalindromo("La ruta nos aporto otro paso natural"); // debe retornar Es palíndromo
console.log(esPalindromo("La ruta nos aportol"); // debe retornar No es palíndromo

39 - Crea una clase Ficha, con su constructor, que se usará para almacenar el número de kilómetros recorridos por una persona en cada sesión de ejercicios. Las propiedades de la clase serán

- nombre
- sesiones
- numsesiones

El nombre es el de la persona, en las sesiones se almacenará el número de kilómetros recorridos en cada sesión y numsesiones contiene el número de sesiones anotadas.

Tiene dos métodos:

- anotar: anota los kilómetros

- media: calcula la media de kilómetros recorridos

Cada persona tendrá su ficha construída con esta clase.
Si ejecuto anotar(8), anotar(10), anotar(6), en las sesiones se anotarán 8, 10, 6.
Si escribo media() devolverá 8 ((8+10+6)/3).

40 - Crea una clase que llamaremos Colectivo. Los atributos de la misma serán:

- capacidad: número máximo de pasajeros
- pasajeros: número de pasajeros (inicialmente 0)
- conductor: objeto conductor.

Sus métodos

- subir(pasajeros): aumenta el número de pasajeros
- bajar(pasajaeros): disminuye el número de pasajeros
- conductor: asigna un objeto conductor.

El ojeto conductor es de una clase (Conductor) cuyos atributos son:

- nombre: nombre del conductor
- licencia: un número que identifica al conductor.

Al crear el objeto se asigna también el conductor

No pueden subir más pasajeros que los máximos admitidos y no pueden bajar más de los que hay.

El autobús linea1 puede llevar 40 pasajeros y su conductor se llama José su licencia es la 1234.

- ☐ Si se pide subir(25) , el atributo pasajeros valdrá 25.
- ☐ Si a continuación se pide subir(35) solo subirán 15, (25 + 15 son los 40 de máximo)
- ☐ Si pedimos bajar 45, el autobús se queda vacío.
- ☐ Si teniendo 35 pasajeros se pide que bajen 40 el autobús se queda vacío.