Ejercicios nivelatorios para el Coding Bootcamp.

¡Hola! ¿Qué tal? Te presentamos en esta oportunidad los 21 ejercicios que debés presentar para poder habilitar la fecha de tu Challenge.

Recordá que:

* No podés editar el documento actual (copialo y pegalo en tu documento).
* Los ejercicios deben estar resueltos en lenguaje JavaScript.
* Debés entregarlos adjuntos dentro del formulario <https://bit.ly/P5FormularioDeAplicacion> en formato PDF.
* Deben estar bien resueltos los ejercicios para poder avanzar a la instancia del challenge.

¡Mucha suerte!

### **1. *Loop* de pares**

Debés crear una función llamada loopDePares que reciba como parámetro un número y haga un *loop* de 0 a 100, mostrando en la consola cada número del *loop*.   
En caso de que el número de la iteración, sumado con el número pasado por parámetro, sea par, mostrá en la consola “El número x es par”.

### **2. *Loop* de impares con palabra**

Debés crear una función llamada loopDeImpares que reciba como parámetro un número y una palabra y haga un *loop* de 0 a 100, mostrando en la consola cada número del *loop*.

Ahora, modificá el código para que, en caso de que ese número sumado con el número pasado por parámetro sea impar, muestre en la consola la palabra pasada por parámetro.

### **3. 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

### **4. Callback**

Debes crear una función llamada *callback* que reciba como Parámetros un número y una función. Esta deberá retornar el resultado de esa función pasándole como argumento el número que llega por parámetro.

Teoría de callbacks: [click aqui](https://lenguajejs.com/javascript/asincronia/callbacks/)

Ejemplo:

* callback(5, (num)=>{return num\*10}) debe retornar 50
* callback(25, (num)=>{return num/5}) debe retornar 5

### **5. Nuevo Arreglo**

Debés 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]

### **6. Similar String.split()**

Debés crear una función llamada `split` que reciba un `string` y simule el comportamiento de la función original. Si no sabés cómo funciona, Google puede ayudarte.

Importante: No podés usar el String.split()

Ejemplo:

* split(“hola”) debe retornar [“h”,”o”,”l”,”a”]
* split(“chau”) debe retornar [“c”,”h”,”a”,”u”]

### **7. Carácter del medio**

Debés crear una función llamada `middleCharacter` que reciba un `string` por parámetro y devuelva sus caracteres del medio.

Ejemplo:

* middleCharacter(“plataforma5”) debe retornar “f”
* middleCharacter(“hola”) debe retornar “ol”
* middleCharacter(“cosas”) debe retornar “s”

### **8. Mover ceros a lo último**

Debés 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]

### **9. Manejar dos arreglos**

Debés crear una función llamada `arrayHandler` que reciba dos arreglos de igual largo como parámetros y muestre en la consola “Soy {elemento de array 1} y yo soy {elemento de array 2}”.

Ejemplo:

* arrayHandler([1,2,3,4], [“h”,”o”,”l”,”a”]) debe mostrar:

Soy 1 y yo soy h

Soy 2 y yo soy o

Soy 3 y yo soy l

Soy 4 y yo soy a

### **10. Mezclar arreglos**

Debés crear una función llamada `mezclarArreglos` que reciba dos arreglos como parámetros y devuelva uno nuevo con los elementos de ambos, de 1 en 1.

Ejemplo:

* mezclarArreglos([1,2,3,4],[“h”,”o”,”l”,”a”]) debe retornar [1, “h”, 2,”o”, 3,”l”,4,”a”]
* mezclarArreglos([1,2,3,4], [“h”,”p”]) debe retornar [1, “h”, 2, “p”, 3 , 4]
* mezclarArreglos([“h”,”p”], [1,2,3,4]) debe retornar [“h”, 1, “p”, 2, 3 , 4]

### **11. Arreglos**

Investigá los siguientes métodos de los arreglos y replicá su funcionamiento de manera lógica al igual que lo hiciste en el ejercicio número 5:

* 1. “join([1,2,3,4,5], “ ”)” debe retornar “1 2 3 4 5”
* 2. “pop([1,2,3,4,5])” debe retornar 5
* 3. “filter([1,2,3,4,5], (e)=>{return e%2===0})” debe retornar [2, 4]
* 4. “map([1,2,3,4,5], (e)=>{return e \* 10})” debe retornar [10, 20, 30, 40, 50]

### **12. Mínima Suma**

Debés crear una función llamada `minSum` que reciba un arreglo de números desordenados y devuelva la suma entre los dos números más chicos.

Ejemplo:

* minSum([7, 6, 5, 4, 3, 2, 1]) debe retornar 3 (2 + 1)
* minSum([1, 10, 43, 900, 20, 8]) debe retornar 9

### **13. Arreglo de objetos**

Debés crear una función llamada `arregloDeObjetos` que reciba un número como parámetro y devuelva un arreglo de objetos que tengan una propiedad llamada `valor` y 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}]

### **14. Arreglo de objetos**

Debés crear una función llamada `arregloDeObjetos` que reciba un número y una palabra como parámetro y devuelva un arreglo de objetos que tenga: una propiedad llamada como la palabra pasada por parámetro y el valor del número y sus anteriores.

Ejemplo:

* arregloDeObjetos(5, “hola”) debe retornar [{hola: 1}, {hola: 2}, {hola: 3}, {hola: 4}, {hola: 5 }]
* arregloDeObjetos(3, “chau”) debe retornar [{chau: 1}, {chau: 2}, {chau: 3}]

### **15. Única propiedad**

Debés crear una función llamada `oneProperty` que reciba un arreglo de objetos como parámetro y un string. Deberá retornar un nuevo arreglo de objetos, teniendo como propiedad al string pasado como parámetro.

Ejemplo:

*var arreglo = [ { name: “lucas”, edad: 20 }, { name: “santi”, edad: 22 } ]*

* oneProperty(arreglo, “edad”) debe retornar [ { edad: 20 }, { edad: 22 } ]
* oneProperty(arreglo, “name”) debe retornar [ { name: “lucas”}, { name: “santi” } ]

### **16. Guerra de palabras**

Utils:

*var abc = {a:1,b:2,c:3,d:4,e:5,f:6,g:7,h:8,i:9,j:10,k:11,l:12,m:13,n:14,o:15,p:16,q:17,r:18,s:19,t:20,u:21,v:22,w:23,x:24,y:25,z:26}*

Debés crear una función llamada `warWords` que reciba dos palabras como parámetro y devuelva la ganadora según la suma del valor de sus letras (dado por su posición en el abecedario).

Ejemplo:

* warWord(“hola”, “chau”) debe retornar “hola” (36 > 33)
* warWord(“love”, “friendship”) debe retornar “friendship”

### **17. Prefijos Telefónicos**

Utils:

*var prefijos = [54, 55, 56, 57, 58]*

*var paises = ["argentina", "brasil", "chile", "colombia", "venezuela"]*

Debés crear una función llamada `validarPrefijo` que reciba un número por parámetro. Deberá:

1. Generar un objeto desde dos arreglos dados. Deberá tener como propiedades, los números de prefijos, y como valor, el país correspondiente a cada prefijo.  
   ejemplo :   
     
   {  
    54: “Argentina”,  
    55: ”Brasil”,  
    56: ”Ecuador”,  
    57: ”Bolivia”  
   }
2. Validar si los dos primeros números del pasado por parámetro existen en el objeto de prefijos telefónicos. En caso de ser correcto, deberá retornar “Este número pertenece a X”.
3. En caso de ser incorrecto deberá retornar “El número no pertenece a nuestros países”

Ejemplo:

* validarPrefijo(“5412345678”) debe retornar “Este número pertenece a Argentina”
* validarPrefijo(“5712345678”) debe retornar “Este número pertenece a Bolivia”
* validarPrefijo(“8012345678”) debe retornar “El número no pertenece a nuestros países”

### **18. {value: key} kelue: vay**

Debés crear una función llamada `reverseKeys` que reciba un objeto como paŕámetro. Si ese paŕametro no es un objeto, deberá devolver un string que indique qué tipo de dato fue ingresado como input y por qué no es válido.

Si el parámetro es un objeto, la función deberá devolver otro objeto. Deberá tener todas las propiedades que sean un string puestas como key y el value deberá ser el key anterior.

Ejemplo:

* reverseKeys(344) debería devolver “error, input can’t be a number”
* reverseKeys([ ]) debería devolver “error, input can’t be an Array”
* reverseKeys(‘hola’) debería devolver “error, input can’t be a string”

var prueba = {

nombre: ‘santi’,

edad: 22,

nacionalidad: ‘de otro planeta’,

documento: 44444444

}

* reverseKeys(prueba) debería devolver:

{

santi: ‘nombre’,

edad: 22,

‘’de otro planeta”: nacionalidad,

documento: 44444444

}

### **19. Palíndromo**

Deben crear una función llamada “palíndromo” que indique si una palabra es palíndroma o no, debe retornar true en el caso que sea, y false en el caso que no.

Definición de palíndromo [aquí](https://es.wikipedia.org/wiki/Pal%C3%ADndromo).

Ejemplo:

* palindromo(“anilina”) debe retornar true
* palindromo(“Ana”) debe retornar true
* palindromo(“Enrique”) debe retornar false

### **20. Rotación de arreglo a la derecha**

Debes crear una función llamada “rotar” que reciba un arreglo y un número, y debera retornar un nuevo arreglo con los elementos rotados la cantidad de veces como sea el número a la derecha.

Ejemplo:

* rotar([1, 2, 3, 4, 5], 2) debe retornar [4, 5, 1, 2, 3]
* rotar([1, 2, 3, 4, 5], 3) debe retornar [3, 4, 5, 1, 2]

### 

### **21. Consecutivos similares**

Debes crear una función llamada “consecutivosSimilares” que reciba un string y retorne cuantos consecutivos similares hay en un string.

Ejemplo:

* consecutivosSimilares(“AAAA”) debe retornar 3
* consecutivosSimilares(“BBBBB”) debe retornar 4
* consecutivosSimilares(“ABABABAB”) debe retornar 0
* consecutivosSimilares(“BABABA”) debe retornar 0
* consecutivosSimilares(“AAABBB”) debe retornar 4

*Nota: Si querés prepararte mejor para el Challenge, podés elegir al azar 5 de estos ejercicios y resolverlos tomándote el tiempo. Sería genial que puedas resolverlos en 2 horas máximo.*