TA 1

Dado un archivo HTML, crear un archivo .js que implemente una función llamada **repeatString (texto: string, repeticiones:int), que utilizando un loop, imprima el páramentro texto tantas veces como el parámetro repeticiones indique.**

La impresión puede hacerse usando console.log(texto)

TA 2

Dado un archivo HTML, crear un archivo .js que implemente una función llamada **reverseString**(texto: string), que invierta el párametro texto y lo imprima usando console.log().

Pueden usarse diferentes métodos provistos por JavaScript, la idea no es hacer el reverso a mano

TA 3

Dado un archivo HTML, crear un archivo .js que implemente una función llamada **removeFromArray**(arreglo: Array<any>, item: <any>) que remueva el parámetro item del párametro arreglo, e imprima arreglo usado console.log();

Pueden usarse diferentes métodos provistos por JavaScript.

TA 4

Dado un archivo HTML, crear un archivo .js que implemente una función llamada **sumAll**(a: int, b: int) que calcule la suma de todos los números desde el parámetro a al parámetro b (incluídos) y lo imprima usando console.log()

TA 5

Dado un archivo HTML, crear un archivo .js que implemente una función llamada **leapYears**(año: int), que imprima en consola true si el parámetro año es un año bisiesto y false en caso contrario.

Un año bisiesto es aquel que es divisible entre 4, o si es divisible entre 100 y también es divisible entre 400.

TA 6

Dado un archivo HTML, crear un archivo .js que implemente 2 funciones llamadas **convertToCelsius**(temp: int) y **convertToFahrenheit(**temp: int), que haga la conversión de un valor de temperatura de Celsius a Fahrenheit y viceversa, e imprima los resultados en consola.

Nos interese que el resultado sea legible para una persona, por lo que el resultado debe estar redondeado a un valor decimal (ej.: convertToCelsius(100) debe retornar 37.8 y no 37.77777777777778)

TA 7

Dado un archivo HTML, crear un archivo .js que implemente una función llamada **getTheTitles**(books: Array), que devuelva e imprima en consola un nuevo arreglo que contenga sólamente los títulos de los libros guardados en el parámetro books.

El parámetro books debe tener este formato (puede y debe agregar más libros):

```
const books = [
 {
   title: 'Book',
   author: 'Name'
 },
 {
   title: 'Book2',
   author: 'Name2'
 }
]
```

Se pueden usar métodos provistos por JavaScript para realizar este ejercicio.

TA 8

Dado un archivo HTML, crear un archivo .js que implemente una función llamada **findTheOldest**(people: Array), que devuelva e imprima en consola aquella persona dentro del parámetro people que sea el que tenga más edad.

El parámetro people debe tener este formato (puede y debe agregar más personas):

```
const people = [
    {
      name: "Carly",
      yearOfBirth: 1942,
      yearOfDeath: 1970,
    },
```

```
  {
    name: "Ray",
    yearOfBirth: 1962,
    yearOfDeath: 2011,
  },
  {
    name: "Jane",
    yearOfBirth: 1912,
    yearOfDeath: 1941,
  },
]
```

Se pueden usar métodos provistos por JavaScript para realizar este ejercicio.

TA 9

Dado un archivo HTML, crear un archivo .js que implemente una función llamada **getOdds**(nums: Array<int>) que filtre e imprima en consola, en un nuevo arreglo, aquellos números dentro del parámetro nums que sean impares.

No se debe iterar manualmente sobre el parámetro nums, utilice métodos provistos por JavaScript para realizar este ejercicio.

TA 10

Dado un archivo HTML, crear un archivo .js que implemente una función llamada **getSum**(nums: Array<int>) que calcule la suma de todos los numéros dentro del parámetro nums y la imprima en consola.

No se debe iterar manualmente sobre el parámetro nums, utilice métodos provistos por JavaScript para realizar este ejercicio.

TA 11

Dado un archivo HTML, crear un archivo .js que implemente una función llamada **duplicates**(nums: Array<int>) que calcule la cantidad de elementos repetidos dentro del parámetro nums y la imprima en consola.

Ej.: [1, 2, 2, 3, 4, 4, 4, 5] debería devolver 2 (ya que el 2 y el 4 están duplicados)

TA 12

Dado un archivo HTML, crear un archivo .js que implemente una función llamada **generatePassword**(length: int) la cual debe crear una contraseña de largo igual al parámetro length, y que debe cumplir lo siguiente:

- La contraseña debe incluir letras mayúsculas, minúsculas, números y símbolos especiales
- Debe tener un largo mínimo de 8 caracteres
- Llamar, por ejemplo, **generatePassword**(8) dos veces, debería producir resultados diferentes

TA 13

Objetivo: Cambiar el texto de un elemento al hacer clic en un botón.

Instrucciones:

- Crear un archivo HTML con un elemento <p> que tenga el texto "Texto original".
- Añadir un botón con el texto "Cambiar texto".
- Al hacer clic en el botón, el texto del párrafo debe cambiar a "Texto cambiado".

TA 14

Objetivo: Añadir nuevos elementos a una lista al hacer clic en un botón.

Instrucciones:

- Crear una lista <ul> vacía en un archivo HTML.
- Añadir un input y un botón que diga "Añadir elemento".
- Al escribir texto en el input y hacer clic en el botón, se debe añadir un nuevo <li> con el texto ingresado a la lista, y el texto del input debe borrarse

TA 15

Objetivo: Eliminar un elemento específico de la lista.

Instrucciones:

- Usando una copia del HTML anterior, agregar un botón "Eliminar último elemento".
- Al hacer clic en el botón, se debe eliminar el último <li> de la lista.

TA 16

Objetivo: Mostrar y ocultar un elemento al hacer clic en un botón.

Instrucciones:

- Crear un archivo HTML con un elemento <p>, con un texto cualquiera.
- Añadir un botón que diga "Mostrar/Ocultar".
- Al hacer clic en el botón, el párrafo debe ocultarse si está visible y mostrarse si está oculto

TA 17

Objetivo: Crear un contador que aumente cada vez que se hace clic en un botón.

Instrucciones:

- Crear un archivo HTML con un elemento <p> con el número 0.
- Añadir un botón con el texto "Incrementar".
- Al hacer clic en el botón, el número dentro del debe incrementarse en uno.

TA 18

Objetivo: Filtrar elementos de una lista mientras se escribe en un input.

Instrucciones:

- Crear un archivo HTML con una lista <ul>, de personas. Dicha lista debe ser renderizada de forma dinámica desde un archivo JavaScript usando template strings
- Añadir un input de búsqueda.
- Mientras el usuario escribe en el input, los elementos de la lista que no coincidan con el texto deben ocultarse.

TA 19

Objetivo: Validar un formulario en tiempo real.

Instrucciones:

- Crear un formulario con campos de texto para nombre, contraseña y correo electrónico.
- Añade un mensaje de error debajo de cada campo que aparezca si los campos están vacíos, la contraseña tiene menos de 8 caracteres, o si el correo no es válido.
- El formulario no debe enviarse si hay errores.

TA 20

Objetivo: Cambiar la imagen al pasar el mouse por encima y volver a la original al quitarlo.

Instrucciones:

- Colocar una imagen en tu página.
- Usar algún evento para cambiar la imagen a una diferente cuando el mouse pase por encima.
- Usar algún otro evento para volver a la imagen original cuando el mouse se retire.

TA 21

Objetivo: Cambiar el estilo de un campo de texto cuando está enfocado.

Instrucciones:

- Crear un input de texto.
- Usar algún evento para cambiar el borde del input a un color más oscuro cuando el campo está activo
- Usar algún otro evento para restaurar el borde original cuando el campo pierde el foco.

TA 22

Objetivo: Detectar el cambio de tamaño de la ventana del navegador.

Instrucciones:

- Usar algún evento para detectar cuando el usuario cambia el tamaño de la ventana.
- Mostrar el tamaño actual de la ventana en un párrafo dentro de la página.