Desarrollo Web en Entorno Cliente

UT1. Javascript Actividades

Actualizado Septiembre 2022

ÍNDICE DE CONTENIDO

1. EJERCICIOS BÁSICOS

Guarda cada ejercicio en un .js con un nombre descriptivo:

1. Dados dos números indicar cuál es mayor, menor o si son iguales.
2. Modifica el programa anterior: si los números no son un número o son menores o iguales a ceros, que los vuelva a pedir.
3. Dada una hora en horas, minutos y segundos, indicar qué hora será pasado un segundo.
4. Utilizando un bucle, mostrar la suma, el producto y la media de los números introducidos hasta introducir un número negativo y entonces mostrar el resultado.
5. Mostrar todos los números que hay entre dos números introducidos por el usuario.
6. Mostrar todos los números impares que hay entre dos números introducidos por el usuario.
7. Mostrar todos los números divisores de un número introducido por el usuario.
8. Mostrar la tabla de multiplicar de un número introducido por pantalla.
9. Realizar una pequeña calculadora, donde el programa solicite dos números y una operación aritmética simple (+,-,*,/). El programa debe validar que los datos introducidos por el usuario son correctos. Si no lo son, solicitarlos de nuevo, si lo son, mostrar el resultado.
10. Programa una función que determine si un número es primo (aquel que solo es divisible por sí mismo y 1) o no, pe. miFuncion(7) devolverá true
11. Programa una función que determine si un número es par o impar, pe. miFuncion(29) devolverá Impar.
12. Programa una función para convertir grados Celsius a Fahrenheit y viceversa, pe. miFuncion(0,"C") devolverá 32°F
13. Programa una función que calcule el factorial de un número (El factorial de un entero positivo n, se define como el producto de todos los números enteros positivos desde 1 hasta n), pe. miFuncion(5) devolverá 120.
14. Programa una función que devuelva el monto final después de aplicar un descuento a una cantidad dada, pe. miFuncion(1000, 20) devolverá 800
15. Indica si un NIF es válido o no.
16. Crea una función para dibujar un patrón de diente de sierra inverso en un cuadro de texto. Con un carácter y un número que indique el mayor número de caracteres en la base (inversa) del patrón.
    Ejemplo 1. Datos de entrada: 'A' y 5
    AAAAA
    AAAA
    AAA
    AA
    A
17. Programa una función que cuente el número de caracteres de una cadena de texto, pe. miFunción("Hola Mundo") devolverá 10.
18. Programa una función que te devuelva el texto recortado según el número de caracteres indicados, pe. miFunción("Hola Mundo", 4) devolverá "Hola".
19. Programa una función que dada una String te devuelva un Array de textos separados por cierto carácter, pe. miFuncion('hola que tal', ' ') devolverá ['hola', 'que', 'tal']
20. Programa una función que repita un texto X veces, pe. miFuncion('Hola Mundo', 3) devolverá Hola Mundo Hola Mundo Hola Mundo.

21. Programa una función que invierta las palabras de una cadena de texto, pe. miFuncion("Hola Mundo") devolverá "odnuM aloH".
22. Programa una función para contar el número de veces que se repite una palabra en un texto largo, pe. miFuncion("hola mundo adios mundo", "mundo") devolverá 2.
23. Programa una función que valide si una palabra o frase dada, es un palíndromo (que se lee igual en un sentido que en otro), pe. mifuncion("Salas") devolverá true.
24. Programa una función que elimine cierto patrón de caracteres de un texto dado, pe. miFuncion("xyz1, xyz2, xyz3, xyz4 y xyz5", "xyz") devolverá "1, 2, 3, 4 y 5.
25. Programa una función que reciba un número y evalúe si es capicúa o no (que se lee igual en un sentido que en otro), pe. miFuncion(2002) devolverá true.
26. Comprueba que una cadena empieza con las letras “m” o “d” y además termina con las letras “a” o “o”. Realiza el ejercicio con funciones de cadena y con expresiones regulares.
27. En un vector de números, indicar:
    a. El número de elementos del vector.
    b. Cuántos son pares y cuántos impares y cuáles son.
    c. La suma de todos los números negativos.
    d. El producto de todos los números positivos.
    e. Cuántos son primos y cuáles son.
    f. Los números que ocupan las posiciones pares del vector.
    g. El número mayor.
    h. El número menor.
    i. La media de todos los números, los números que están por encima y los que están por debajo.
    j. El vector ordenado de mayor a menos y viceversa.
    k. Buscar un valor introducido por el usuario e indicar si existe o no.
28. En un vector de cadenas, indicar:
    a. La cadena más corta.
    b. La cadena más larga.
    c. La cadena con más letras ‘a’.
    d. Cuántas cadenas hay que tengan la ‘b’ y cuáles son.
29. Programa una función para convertir números de base binaria a decimal y viceversa, pe. miFuncion(100,2) devolverá 4 base 10
30. Programa una función para saber la edad de una persona, sabiendo la fecha de nacimiento.
31. Programa una función que dada una fecha válida determine cuantos años han pasado hasta el día de hoy, pe. miFuncion(new Date(1984,4,23)) o miFuncion (“01/12/2010”)
32. Programa una función que dada una cadena de texto cuente el número de vocales y consonantes, pe. miFuncion("Hola Mundo") devuelva Vocales: 4, Consonantes: 5
33. Programa una función que valide que un texto sea un nombre válido, p.e. miFuncion (“Javier Ferrer”) devolverá verdadero. NOTA: No puede haber números ni caracteres especiales como ¡ o ¿
34. Programa una función que valide que un texto sea un email válido, p.e. miFuncion (“javier.ferrer@iesmartinezm.es”) devolverá verdadero. NOTA: caracteres, números, puntos guión alto y bajo+@+caracteres, números+.+al menos dos caracteres.
35. Programa una función que dado un array numérico devuelve otro array con los números elevados al cuadrado, pe. mi_funcion([1, 4, 5]) devolverá [1, 16, 25]
36. Programa una función que dado un array devuelva el número más alto y el más bajo de dicho

array, p.e. miFuncion([1,5,34,99,-2]) devolverá [99,-60]

37. Programa una función que dado un array de números devuelva un objeto con 2 arreglos en el primero almacena los números pares y en el segundo los impares, pe. miFuncion([1,2,3,4,5,6,7,8,9,0]) devolverá {pares: [2,4,6,8,0], impares: [1,3,5,7,9]}
38. Programa una función que dado un arreglo de números devuelva un objeto con dos arreglos, el primero tendrá los numeros ordenados en forma ascendente y el segundo de forma descendiente, pe. miFuncion([7, 5,7,8,6]) devolverá { asc: [5,6,7,7,8], desc: [8,7,7,6,5] }
39. Programa una función que dado un array de elementos, elimine los duplicados, pe.e mi Funcion ("x",10,"x",2,"10,10, true,true) devolverá ["x",10,2,"10",true]
40. Programa una función para devolver la edad de una persona dada su fecha de nacimiento en este formato dd/mm/aaa.
41. Programa una función que obtenga un numero aleatorio entre 501 y 600.
42. Programa una clase llamada Pelicula.

La clase recibirá un objeto al momento de instanciarse con los siguientes datos: id de la película en IMDB, titulo, director, año de estreno, país o países de origen, géneros y calificación en IMBD.

- Todos los datos del objeto son obligatorios.
- Valida que el id IMDB tenga 9 caracteres, los primeros 2 sean letras y los 7 restantes números.
- Valida que el título no rebase los 100 caracteres.
- Valida que el director no rebase los 50 caracteres.
- Valida que el año de estreno sea un número entero de 4 dígitos.
- Valida que el país o paises sea introducidos en forma de arreglo.
- Valida que los géneros sean introducidos en forma de arreglo.
- Valida que los géneros introducidos esten dentro de los géneros aceptados*.
- Crea un método estático que devuelva los géneros aceptados*.
- Valida que la calificación sea un número entre 0 y 10 pudiendo ser decimal de una posición.
- Crea un método que devuelva toda la ficha técnica de la película.
- Apartir de un arreglo con la información de 3 películas genera 3 instancias de la clase de forma automatizada e imprime la ficha técnica de cada película.

* Géneros Aceptados: Action, Adult, Adventure, Animation, Biography, Comedy, Crime, Documentary ,Drama, Family, Fantasy, Film Noir, Game-Show, History, Horror, Musical, Music, Mystery, News, Reality-TV, Romance, Sci-Fi, Short, Sport, Talk-Show, Thriller, War, Western.

2. PIEDRA, PAPEL O TIJERA

Realiza el juego de piedra, papel o tijera contra otro jugador o contra la máquina (con números aleatorios)

3. TRES EN RAYA

Realiza el juego del tres en raya contra otro jugador o contra la máquina (con números aleatorios). Puedes hacerlo usando en cada jugada las coordenadas de X u O desde 0,0 (esquina superior izquierda) a 2,2 (esquina inferior derecha).

4. SUDOKU

Realiza una aplicación web que compruebe si una solución de un Sudoku es correcta o no.
Una sugerencia de diseño para la aplicación web es que debe tener 9x9 campos de texto y un botón "Comprobar". Recomiendo un valor por defecto de un Sudoku válido para hacer pruebas

La explicación de las reglas del Sudoku las tenéis aquí https://es.wikipedia.org/wiki/Sudoku
Internamente el programa realizará la comprobación de si el Sudoku es o no correcto en una función definida como
function esSudokuCorrecto(miArrayBi)
que devolverá true si es correcto, false en caso contrario.

5. NÚMERO PATRONES

Realiza una aplicación web que solicite una cadena de texto.
El programa debe decir cuántas veces ocurre cada uno de estos patrones sin distinguir mayúsculas y minúsculas: “00” “101”, “ABC”, “HO”.
Un carácter puede formar parte de más de un patrón encontrado. Por ejemplo:
En la cadena “000” el patrón “00” aparece dos veces (una empieza en la posición 0 y otra empieza en la posición 1).
Internamente el programa realizará la cuenta de patrones con una función definida como
function numeroPatrones(texto)
que devolverá un número entero con el número de patrones encontrados.

6. BUSCAMINAS

Realiza una aplicación web que reciba en código mediante un array bidimensional (de longitud variable) un escenario de Buscaminas, donde haya un 0 donde no hay minas y un -1 donde si hay. Para cada casilla que no tenga una mina, diga cuantas minas adyacentes hay (en diagonal, horizontal y vertical).
Internamente el programa realizará las acciones con una función definida como
function contandoMinas(miCampo)
que devolverá un array bidimensional con el número de minas adyacentes en cada posición. Esta
Más información de cómo funciona el Buscaminas
https://es.wikipedia.org/wiki/Buscaminas
Ejemplo Entrada
0 0 -1 0
0 -1 -1 0
Ejemplo Salida
1 3 -1 2
1 -1 -1 2

7. AGENDA

Crea un fichero HTML básico que cargue el siguiente código javascript:

```
const inicioDeJornada = "07:30";
const finalDeJornada = "17:45";
function agendarReunión(horaDeInicio,duracionEnMinutos) {
    //ToDo...
    return true;
}
console.assert(agendarReunión("7:00",15)==false, 'Fallo comprobando
agendarReunión("7:00",15)==false');
console.assert(agendarReunión("07:15",30)==false, 'Fallo comprobando
agendarReunión("07:15",30)==false');
console.assert(agendarReunión("7:30",30)==true, 'Fallo comprobando
agendarReunión("7:30",30)==true');
console.assert(agendarReunión("11:30",60)==true, 'Fallo comprobando
agendarReunión("11:30",60)==true');
console.assert(agendarReunión("17:00",45)==true, 'Fallo comprobando
agendarReunión("17:00",45)==true');
console.assert(agendarReunión("17:30",30)==false, 'Fallo comprobando
agendarReunión("17:30",30)==false');
console.assert(agendarReunión("18:00",15)==false, 'Fallo comprobando
agendarReunión("18:00",15)==false');
console.assert(agendarReunión("15:30",180)==false, 'Fallo comprobando
agendarReunión("15:30",180)==false');
console.log("Test Done!");
```

Rellena el contenido de la función "**agendarReunión**" de modo que devuelva true si la reunión ocurre dentro del horario laboral y false en caso contrario.
Idea una estrategia para poder trabajar con los distintos tipos básicos involucrados.
Los assert del final te indicarán si tu solución es correcta.

6. CLASE COLEGIO

Diseña una clase “Colegio”. Dicha clase tendrá como atributos “nombre”, numeroAulas” y “numeroAlumnos”. Cada alumno se representará como un objeto de la clase “Alumno”. En ella se guardaran los atributos “DNI”, “nombre” y “notaMedia”.
Implementa métodos en Colegio y Alumno para modificar la nota media.
Verifica que funcione correctamente con un par de ejemplos.

7. CLASE AEROPUERTO

Diseña una clase “Aeropuerto”. Tendrá como atributos “nombre”, “ciudad” y “numeroVuelosDiarios”. Cada vuelo diario se representará como una objeto de la clase “Vuelo”. En ella se guardaran los atributos “codigo”, “hora_llegada” y “hora_salida”.
Implementa métodos en aeropuerto y vuelo para modificar la hora de llegada, para modificar la hora de salida y para comprobar si la hora de salida es posterior a la hora de llegada.
Verifica que funcione correctamente con un par de ejemplos.

8. CLASE HOSPITAL

Diseña una clase “Hospital”. Tendrá como atributos “nombre”, “ciudad” y “numPacientes” (número

de pacientes ingresados). Cada paciente se representará como un objeto de la clase "Paciente". En ella se guardaran los atributos "DNI", "nombre", "enfermedad".
Implementa un método en "Hospital" que reciba el código de paciente y se le de alta a dicho paciente (equivale a eliminar al paciente).

9. TEMPORIZADOR DNI

Realiza un programa que cada 20 segundos (mediante setInterval) solicite un DNI hasta que alguien le escriba la cadena "-1".
En ese momento, el programa debe mostrar seguidas las letras de todos los DNIs introducidos.
Aquí un enlace para saber como calcular la letra de DNI:
https://es.wikipedia.org/wiki/N%C3%BAmero_de_identificaci%C3%B3n_fiscal

10. TEMPORIZADOR FECHA

Realiza un programa que pasados 20 segundos, nos muestre una sola vez la fecha actual del sistema.

11. DNIS POR LETRA

Realiza un programa que pregunte una letra de la A a la Z. Tras ello el programa indicará cuántos DNIs de 3 cifras (del 001 al 999) tienen esa letra y tras ello te mostrará "de golpe" el listado de todos los DNIs que tienen esa letra.

12. PRIMOS Y PALÍNDROMOS

Realiza un programa que calcule cuántos números son a la vez primos y palíndromos desde el 1 hasta 100000. Debe guardar todos ellos en un array y al finalizar el proceso imprimir dicho array.
Definición de número primo:
https://es.wikipedia.org/wiki/N%C3%BAmero_primo
Definición de palíndromo:
https://es.wikipedia.org/wiki/Pal%C3%ADndromo

13. DESGLOSE CADENA

Realiza un programa que reciba una cadena con el siguiente formato:
"nombre:apellidos:telefono:email:codigopostal"
Tras recibir la cadena, debe desglosar y mostrar la siguiente información:

- Código postal.
- Apellidos.
- Email.
- Suponiendo un formato de email "direccion@servidor" debe mostrar el nombre del servidor asociado.