Programación básica

Ejercicio 1: Añadir a una pagina HTML un script que muestre un `alert` de la siguiente forma:

a) Añadimos el `alert` de forma interna sin función.
b) Añadimos el `alert` en un evento en un botón.
c) Igual que el punto anterior pero usamos una función.
d) Igual pero la función en un archivo externo.

Ejercicio 2: Crear una página con una imagen y un botón. Añadir el código necesario para que cuando pulse el botón salga un `alert ("hola")` y cuando pase el ratón encima de la foto *(onmouseover)* salga un `alert("adios")`.

Ejercicio 3: Crear una página web con dos botones. Uno hace un alert normal, y el otro hace un `document.write` ¿Que sucede al usar document.write?.

Ejercicio 4: A través de un prompt, pide el nombre al usuario y saludalo con un alert de la siguiente forma: *"Bienvenid@ a mi página XXXXXX"* (siendo XXXXXX el nombre que ha introducido el usuario).

Ejercicio 5: Pedir dos valores numéricos con prompt y mostrar su suma, resta, multiplicación, división y resto.

Ejercicio 6: Crea un script que pida al usuario un numero entero positivo N mayor a 0. Hay que controlar que el numero introducido sea correcto. Si no es así se volverá a pedir.

A continuación debe realizar lo siguiente:

a) Calcular los divisores del numero N y mostrarlos.
b) Calcular la suma de los cuadrados de los divisores obtenidos en el paso anterior y mostrarla.
c) Indicar si esa suma es un cuadrado o no (con una frase por pantalla)

Nota: NO se pueden usar arrays en este ejercicio.

Ejercicios de Arrays

Ejercicio 7: Crea una función que reciba un array de valores enteros (positivos y negativos), y devuelva otro array con la suma parcial de cada elemento del array que se pasa como parámetro.

Nota: La suma parcial de un elemento del array será la suma de ese elemento y todos los anteriores a él.

Ej: Para el array [1,2,3,4,5,6], el array resultante será [1,3,6,10,15,21]

Ejercicio 8: Crea una función llamada `countBy` la cual recibirá dos números enteros positivos: X e Y. Esta función debe devolver lo siguiente:

- Si alguno de los números es negativo devolverá un array vacío.
- Sino, devolverá con array con los Y primeros múltiplos de X empezando por el 1 (incluido)

Ejercicio 9: Crea un script que pida al usuario un numero entero positivo N mayor a 2. Hay que controlar que el numero introducido sea correcto. Si no es así se volverá a pedir.

A continuación debe crearse una matriz NxN rellena con los resultados de la tabla de multiplicar de N (empezando en 1). Finalmente, muestra por consola la matriz "en forma de matriz"

Ej: Para el número 3, el programa debe mostrar

```
03 06 09
12 15 18
21 24 27
```

Ejercicio 10: Escriba una función que reciba dos arrays y devuelva un nuevo array con los elementos que solo aparecen una vez en total (ya sea en el primer o en el segundo array). El orden debe ser: primero los que están en el primer array y luego los que están en el segundo.

Ejemplos:

```
[1, 2, 3, 3] y [3, 2, 1, 4, 5]) ==> [4, 5]
["Ray", "Jose", "Dani"] y ["Dani", "Jose", "Ivan"]) ==> ["Ray","Ivan"]
[77, "ciao"] y [78, 42, "ciao"]) ==> [77, 78, 42]
[1, 2, 3, 3] y [3, 2, 1, 4, 5, 4]) ==> [4,5]
```

Ejercicio 11: El ácido desoxirribonucleico, ADN, es la principal molécula de almacenamiento de información en los sistemas biológicos. Está compuesto por cuatro bases de ácido nucleico: guanina ("G"), citosina ("C"), adenina ("A") y timina ("T").

El ácido ribonucleico, ARN, es la principal molécula mensajera de las células. El ARN difiere ligeramente del ADN en su estructura química y no contiene timina. En el ARN, la timina se sustituye por otro ácido nucleico, el uracilo ('U').

Cree una función que traduzca una cadena dada de ADN a ARN.

Por ejemplo:
Si se introduce la cadena `GCAT`, *a salida debe ser* `GCAU`
Si se introduce la cadena `GCATCGTA`, *a salida debe ser* `GCAUCGUA`

La cadena de entrada puede tener una longitud arbitraria, incluso puede estar vacía. Se garantiza que toda cadena de entrada es válida, es decir, que cada cadena de entrada sólo estará formada por 'G', 'C', 'A' y/o 'T' en cualquier orden.

Ejercicio 12: Crea un script que filtre un alista de nombres y devuelva otra lista solo con los que son amigos tuyos.
Como eres una persona muy "especial", tú solo eres amigo/a de aquellas personas cuyo nombre se componga exactamente de 4 letras.

Ejemplo:

```
Entrada={Luis", "Estela", "Ángel", "Enya", "Jose Antonio"}
Salida = {"Luis", "Enya"}

Entrada = {"Joaquín", "Manuel", "Carlos"}
Salida = {}
```

Suponemos que los array son correctos y tienen nombres.
Importante: hay que respetar el orden de los nombres en la salida.

Ejercicio 13: Crea una página web con un solo botón en el que, al pulsarlo, se le pida al usuario un año (pej: 1492). A continuación , el programa debe mostrar a través de una ventana modal a qué siglo pertenece el año introducido.

El siglo I abarca desde el año 1 hasta el año 100 inclusive, el siglo II desde el año 101 hasta el año 200 inclusive, etc.

Ejemplos:

```
1705 --> 18
1900 --> 19
1601 --> 17
2000 --> 20
2742 --> 28
```

Importante: intenta que la función que programes tenga el menor número de lineas posibles. MAXIMO 5 lineas contando llaves (y es mucho). Piensa bien.

Ejercicio 14: Define un array con los siguientes colores: `red`, `yellow`, `green`, `white`, `blue`, `brown`, `pink` y `black`. A continuación crea *un generador aleatorio de banderas*:

1. Se pide el numero de franjas de va a tener la bandera (entre 1 y 5).
2. El programa obtiene de forma aleatoria 5 colores del array.
3. Usando `document.write`, crea una tabla de una fila y tantas columnas como colores tenga la bandera generada. Usa el atributo style para rellenar el fondo de cada celda del color adecuado.

a) En el paso 2 se pueden repetir colores en la bandera.
b) en el paso 2 NO se pueden repetir colores en la bandera.
c) En el paso 2 se pueden repetir colores mientras no sean consecutivos (es decir, no puede haber dos franjas juntas con el mismo color)

Ejercicio 15: Usando el array de colores del ejercicio anterior, crea un script que solicite 8 palabras al usuario y las almacene en otro array.
Ordena ese array (el del usuario) de forma que si aparecen colores del array de colores, estos queden al principio del array y el resto de palabras al final. Muestralo por consola.

Ejemplo:

Array de palabras del usuario:

```
casa blue green orden brown bombilla bici pink
```

Array resultante:

```
blue green brown pink casa orden bombilla bici
```

Ejercicio 16: Crea una función llamada `likes` la cual va a recibir como único parámetro un array de nombres. Esta función debe devolver una cadena siguiendo el patrón que se muestra en el ejemplo siguiente:

```
[]                                --> "no one likes this"
["Peter"]                         --> "Peter likes this"
["Jacob", "Alex"]                 --> "Jacob and Alex like this"
["Max", "John", "Mark"]           --> "Max, John and Mark like this"
["Alex", "Jacob", "Mark", "Max"] --> "Alex, Jacob and 2 others like this"
```

Nota: para este ejercicio no se permiten usar funciones de arrays u operadores especiales de Javascript.

Ejercicio 17: Realiza un script que pida 10 números por teclado y que los almacene en un array. A continuación muestra el contenido de ese array junto al índice de cada numero (mira el dibujo). Usa la consola para ello.

Seguidamente el script pedirá dos posiciones a las que llamaremos *inicial* y *final*. Debes comprobar que *inicial* es menor que *final* y que ambos números están entre 0 y 9. Si no es así, vuelve a pedirlos.

A continuación, coloca el número de la posición inicial en la posición final, rotando el resto de números para que no se pierda ninguno.

Al final se debe mostrar el array resultante.

Por ejemplo, para inicial = 3 y final = 7:

Índice	0	1	2	3	4	5	6	7	8	9
Valor	20	5	7	4	32	9	2	14	11	6

Array inicial

Índice	0	1	2	3	4	5	6	7	8	9
Valor	6	20	5	7	32	9	2	4	14	11

Array final

Ejercicio 18: Un restaurante nos ha encargado una aplicación para colocar a los clientes en sus mesas. En una mesa se pueden sentar de 0 (mesa vacía) a 4 comensales (mesa llena).

El funcionamiento es el siguiente:
Cuando llega un cliente se le pregunta cuántos son. Como el programa no está preparado para colocar a grupos mayores a 4, si un cliente solicita una mesa con mas comensales (pej, 6), el programa dará el mensaje “Lo siento, no admitimos grupos de 6, haga grupos de 4 personas como máximo e intente de nuevo” y volverá a preguntar.

Para cada grupo nuevo que llega, se busca siempre la primera mesa libre (con 0 personas). Si no quedan mesas libres, se busca una donde haya hueco para todo el grupo (por ejemplo si el grupo es de dos personas, se podrá colocar en mesas donde haya una o dos personas).

Cada vez que se sientan nuevos clientes se debe mostrar el estado de las mesas.
Los grupos no se pueden romper aunque haya huecos sueltos suficientes.

A tener en cuenta:

- El programa comienza pidiendo el numero de mesas que tiene el restaurante.
- Inicialmente, las mesas se cargan con valores aleatorios entre 0 y 4 y mostrará por pantalla como quedan las mesas inicialmente.
- El programa seguirá pidiendo comensales hasta que se introduzca un valor negativo.

Ejemplo de ejecución:

```
//El usuario ha metido un valor de 10

Estado de las mesas: 3 2 0 2 4 1 0 2 1 1

El usuario pide una mesa para 2.
Por favor, Siéntese en la mesa 3

Estado de las mesas: 3 2 2 2 4 1 0 2 1 1

El usuario pide una mesa para 4
Por favor, Siéntese en la mesa 7

Estado de las mesas: 3 2 2 2 4 1 4 2 1 1
```

```
El usuario pide una mesa para 3
Por favor, Siéntese en la mesa 6

Estado de las mesas: 3 2 2 2 4 4 4 2 1 1

El usuario pide una mesa para 4
Lo siento, no queda sitio en el restaurante.

Estado de las mesas: 3 2 2 2 4 1 4 2 1 1
```

Ejercicio 19: Autómata Celular unidimensional

Supongamos una lista de valores binarios que representan la existencia de células vivas (1) o no (0). Esta lista evoluciona a lo largo del tiempo, creándose nuevas células vivas y/o muriendo otras.

Una celda de la lista evolucionará a partir del estado de sus celdas vecinas (la de su izquierda y la de su derecha) y de ella misma en el instante anterior.

Para ello se siguen las siguientes reglas:

- 000 – 0
- 001 – 1
- 010 – 1
- 011 – 0
- 100 – 1
- 101 – 1
- 110 – 0
- 111 – 0

Una regla `IXD - Y` se interpreta de la siguiente forma: *"La celda X rodeada de las celdas I y D, evolucionará al estado Y"*

Pej:

0**11** – 0 : una celda en el estado 1 rodeada de cero ala izquierda y 1 a la derecha, evolucionará a 0 en el siguiente paso.

1**01** – 1 : una celda en el estado 0 con rodeada de unos, evolucionará a 1 en el siguiente paso.

Vamos a crear un script que nos permita estudiar la evolución del autómata celular antes descrito de tamaño TAM (numero entero mayor o igual a 3) Para ello debes crear las siguientes funciones:

- `inicializarAutomata` : llena el autómata de células muertas.
- `mostrarEstado`: imprime el contenido del autómata teniendo en cuenta que mostrará las células vivas con el carácter '*' y las muertas con '.'
- `primeraJugada`: indica a través de código qué celdas tienen inicialmente células vivas.

Haciendo uso de las funciones anteriores, crea un script que pide al usuario el tamaño del autómata (valor entero mayor o igual a 3) y el número de pasos que va a dar el autómata. Muestra por consola el estado del autómata en cada paso que se vaya dando.

Ejemplo de salida:

Para un autómata de tamaño 15, con la celda 8 como única viva en la primera jugada y 7 pasos, tenemos:

```
Paso 0:. . . . . . . * . . . . . . .
Paso 1:. . . . . . * * * . . . . . .
Paso 2:. . . . . * . . . * . . . . .
Paso 3:. . . . * * * . * * * . . . .
Paso 4:. . . * . . . * . . . * . . .
Paso 5:. . * * * . * * * . * * * . .
Paso 6:. * . . . * . . . * . . . * .
```

Ejercicio 19: Perdido en el bosque

Está en proceso de maduración…

Ejercicio 20: La Búsqueda del Tesoro

Crea un script que simule el siguiente juego:
En un tablero, NxM la maquina colocará de forma aleatoria varias minas y un tesoro. El usuario intentará averiguar la posición del tesoro indicando la posición del tablero que quiere mirar. Podrá seguir jugando mientras no encuentre una de las minas (muere) o el tesoro (gana).

Para este ejercicio usa un tablero de 4x5 con 3 minas.
Hay que asegurarse que tanto las minas como el tesoro no se colocan en posiciones ya ocupadas.
En cada iteración del juego se pintará el tablero y se preguntará al usuario qué coordenadas quiere mostrar del tablero.

Usa la consola para pintar el tablero de la siguiente forma:

- Si una coordenada aún no ha sido visitada, pinta un *
- Si una coordenada ha sido visitada y no tiene nada, pinta un _
- Si una coordenada ha sido visitada y tiene una mina, pinta una X (y mata al jugador)
- Si una coordenada ha sido visitada y tiene el tesoro, pinta un € (y el jugador gana)

Puedes pedir al usuario las coordenadas como quieras: ambos valores de golpe o de uno en uno. Lo único a tener en cuenta es que, para el usuario, las coordenadas empiezan en 1, no en 0.

Mejora: si hay una mina a una casilla de distancia de la posición que ha descubierto el usuario, muestra un aviso indicando que tenga cuidado (pero no reveles la posición de la mina).