Manejando Arrays en JavaScript

NOTA: los objetivos de aprendizaje de esta tarea buscan las siguientes metas:

a) Declarar e invocar funciones

b) Asignar funciones a variables (funciones anónimas)

c) Funciones flecha sin paso de parámetros y con parámetro

d) Ámbito de las variables. Paso por valor y paso por referencia

e) Argumentos de funciones con valor por defecto

f) Número variables de parámetros

g) Recursividad

·Declarar funciones, funciones anónimas y flecha BLOQUE 1·

1. Construye una aplicación web que permita recuperar en una variable el número PI

(Páginas 165, 166, 167, 168 y 169) Se debe implementar la funcionalidad de 3 formas;

1.1. Con una declaración de función tradicional

1.2. Con una función anónima

1.3. Con una función flecha

2. Construye una aplicación web que calcule el área de un rectángulo

(Páginas 165, 166, 167, 168 y 169) Se debe implementar la funcionalidad en base a lo siguiente:

2.1. Se deben recoger los dos valores fuera de la función, está funcionalidad solo se debe añadir una vez.

2.2. Se debe implementar una función para las siguientes formas (hay que tener en cuenta que la función tiene dos pasos de parámetro, uno por la base y otro por la altura del rectángulo):

2.2.1. Declaración de función tradicional

2.2.2. Declaración de función anónima

2.2.3. Declaración de función flecha

2.3. Se debe realizar la llamada a las 3 funciones y se deben mostrar los valores calculados en la pantalla.

·Ámbito de las variables, argumentos, número variables BLOQUE 2·

3. Construye una aplicación web que implemente los siguientes casos.

(Páginas 170, 171, 172, 173 y 174)

3.1. Construye un ejemplo en el que aparezca una variable que se llame igual, y que sea diferente en el programa, una función y dentro de un bucle. En total habría 3 variables.

3.2. Construye un ejemplo que permita crear un array en el programa principal, y sobre este array realice las siguientes acciones (en cada función se debe pasar el array como parámetro, y trabajar dentro de la función la referencia que se ha pasado):

3.2.1. El array debe tener una longitud de 10

3.2.2. Una función que permita rellenar el array con 10 valores aleatorios

3.2.3. Una función que permita ordenar el array. (Se debe gestionar de forma manual, no se pueden utilizar métodos de ordenación)

3.2.4. Una función que permita desordenar el array. (Se debe gestionar de forma manual, no se pueden utilizar métodos de ordenación)

4. Construye una aplicación para probar los argumentos con valores por defecto

(Páginas 173, 174)

4.1. Utilizando los argumentos con valores por defecto crea un programa que realice las siguientes funciones:

4.1.1. El programa va a permitir construir operaciones aleatorias de forma automática.

4.1.2. Existirá una función que devuelva el valor que se ha pasado y en caso de no existirá el valor, devolverá un valor aleatorio (se debe hacer uso de argumento con valores por defecto).

4.1.3. Existirá una función que devuelva el signo de puntuación pasado de entre; + , - , * , / ,y en el caso de que el usuario no pase ningún valor, o no pase uno de estos valores, se deberá generar uno de estos aleatorio (se debe hacer uso de argumento con valores por defecto).

FUNCIONALIDAD DEL SISTEMA

4.1.4. Al arrancar el programa, se le deberá solicitar al usuario, un primer valor numérico, un signo de puntuación y un segundo valor numérico. Estos tres valores se deben solicitar en momentos diferentes (El usuario deberá introducir 3 veces valores).

4.1.5. El usuario puede dejar vacíos cualquiera de estos valores, y en ese caso las funciones anteriores generaran uno aleatorio.

4.1.6. Si el usuario no ha pasado ningún valor, se debe comprobar dentro de cada función respectivamente.

4.1.7. Idea la forma de permitir al usuario introducir el resultado y comprobar si es correcto, haciendo uso de los argumentos con valores por defecto.

5. Construye los ejemplo que se muestran en los siguientes enunciados

(Páginas 174 y 175)

5.1. Crea el ejemplo de calcular la media, haciendo uso de la desestructuración de arrays (páginas 174 y 175)

5.2. Construye una aplicación que visualice los datos de los siguientes vehículos, sabiendo de cual se trata en función del número de parámetros que se le han pasado:

5.2.1. Moto: Cilindrada y peso

5.2.2. Sidecar: Cilindrada, peso y numero pasajeros

5.2.3. Coche: Cilindrada, peso, numero pasajeros y climatizador

5.2.4. Barco: Cilindrada, peso, numero pasajeros, climatizador, radar

5.2.5. Avión: Cilindrada, peso, numero pasajeros, climatizador, radar, piloto automático

5.3. El usuario deberá responder a las preguntas en una sola respuesta, separando cada respuestas por un espacio, u otro carácter que se considere.

5.4. Dependiendo del número de respuestas que haya dado el usuario, se mostrarán los datos que ha introducido el usuario.

·Recursividad BLOQUE 3·

6. Construye el programa de las 8 reinas

(Páginas 180, 181 y 182)

6.1. Construye una aplicación que lleve a cabo la resolución del problema de las 8 reinas, utilizando la recursividad. Utiliza las herramientas que consideres para mostrar el resultado y que se visualicen los pasos que va realizando el programa, permitiendo que estos se vayan realizando de forma rápida, danto tiempo al usuario a ver los cambios entre un cambio y otro.

6.2. Construye un programa que visualice la estructura de las torres de hanoi, y que usando recursividad, resuelva el problema, pasando los discos de uno a otro, y mostrando los pasos que va haciendo al usuario.

·Funciones CallBack BLOQUE 4·

7. Construye los programas que se plantean en los siguientes puntos

(Página 183 y 184)

7.1. Haciendo uso del ejemplo de la página 183 y 184, construye un programa que permita al usuario Escribir un mensaje, y le permita elegir si lo quiere mostrar, con console.error, console.log, alert, prompt y Confirm (Se debe hacer uso de las funciones callBack como en el ejemplo del libro). Páginas 183 y 184

7.2. Construye un programa que realice las cuentas de sumar, restar, multiplicar y dividir haciendo uso de los CallBack (fijarse en el ejemplo de la página 184). La **función principal** debe permitir pasar dos valores numéricos y una función para cada operación.

Dependiendo de si se pasa la función suma, resta, multiplicación y división, la **función principal** debe hacer la llamada a cada una de estas funciones.

Ordenación, recorrido y filtrado de estructuras BLOQUE 5

8. Ordenación de arrays

(Página 185 y 186)

8.1. Construye un programa que permita introducir al usuario 5 acontecimientos históricos y los ordene según su orden cronológico en el que han sucedido. Estos 5 acontecimientos deben estar incluidos dentro de 20 acontecimientos que previamente están grabados en otro array de forma fija en el programa.

8.2. Construye un ejemplo que justifique el uso del método foreach, que sea diferentes al del ejemplo que hemos realizado en clase.

8.3. (Página 187 y 188) Construye un pequeño juego que consista en lo siguiente:

- 8.3.1. Crea un array de 5 elementos, que deben ser 5 valores numéricos.
- 8.3.2. A través del método **map** crea un nuevo array solicitando al usuario que modifique 2 o 3 valores del array (en la llamada al **map**), estos valores adoptaran el doble de lo que valían sus originales. Tener en cuenta que pueden cambiar solo 2 o 3 valores de los 5, y pueden cambiar los de cualquier posición.
- 8.3.3. Un segundo jugador debe intentar adivinar que valores han cambiado.
- 8.3.4. Para ello se deben ir solicitando los valores al usuario primer y al segundo. Se puede elegir el método que se quiera.
- 8.3.5. Los valores se deben ir mostrando en pantalla para hacer un seguimiento del juego.

8.4. (Página 188) Utilizando el método **reduce** construye un programa que calcule el factorial de un número solicitado al usuario.

8.5. (Página 188) Utilizando el método **reduce** construye un programa que calcule la serie de Fibonacci hasta el número que indique el usuario, y lo vaya mostrando, con intervalos de medio segundo.

8.6. (Página 189) Utilizando el método **filter** construye un programa que permita al usuario elegir de entre un menú las siguientes opciones (números de un array del número 1 al 100)

- 8.6.1. Número primos
- 8.6.2. Números pares
- 8.6.3. Números impares
- 8.6.4. Números divisibles entre 3
- 8.6.5. Números divisibles entre 4
- 8.6.6. Números que sumados sean pares.