Programación

Vigésima semana

Marzo 2022

Cruz García, Iago

Anotaciones previas

Estos ejercicios son para familiarizarse con el lenguaje, la sintaxis y cómo resolverlos. Los primeros son sencillos y se va incrementando la dificultad. A continuación se presentan una serie de instrucciones que son necesarias para la resolución de los ejercicios:

- **alert(*parámetro*):** esta instrucción permite mostrar por pantalla un cartel con texto para mostrar la solución de algunos ejercicios.
- **console.log(*parámetro*):** esta instrucción permite mostrar en consola (F12 en el navegador) la solución de algunos ejercicios o trazar el código para comprobar que todo se ejecuta correctamente.
- **prompt(*texto, ejemplo*):** Muestra en pantalla un recuadro de **texto** y un cuadro para introducir texto con un **ejemplo**.
- Para poder ejecutar código JavaScript en Visual Studio Code debéis crear un fichero JavaScript (miScript.js) y un HTML básico (index.html por ejemplo) y dentro de la etiqueta <head> escribir los siguiente:
    - <script src="miScript.js"></script> comillas incluidas
- Ahora que sabemos encapsular creando funciones o métodos, se pueden hacer los ejercicios en el mismo fichero, simplemente comentando las llamadas a métodos que no necesiteis.

```
ejercicio_1()
//ejercicio_2()
//ejercicio_3()
```

Ejercicios

IMPORTANTE: A partir de ahora algunos ejercicios deben hacerse en múltiples ficheros .js, por lo que en vez de entregar todos en un mismo main.js, será necesario dividirlos en directorios. Se aconseja la estructura de "PrácticaX_ejercicio1" y dentro el index.html y los ficheros .js necesarios.

Para estos ficheros, lo mejor es agrupar aquellas funciones o métodos que realicen tareas similares (entradas.js o salidas.js por ejemplo). En caso de duda, no importa que un método quede aislado en un fichero.

El fichero que realice las llamadas a los métodos, que aune toda la funcionalidad, debe nombrarse como main.js y no debe tener más que un método que se llame igual y una llamada a este mismo.

Ejercicio 0

Vamos a hacer unos ejercicios sencillos sobre temporizadores y rutinas, emulando una especie de reloj multifunción muy sencillo, con el que hacer temporizadores o cronómetros con tiempos parciales.

Para empezar, vamos a utilizar el siguiente esquema HTML y JS. Tomaros un tiempo para leerlo, reflexionar sobre él y entender bien cómo funciona.

Os dejo los ficheros compartidos:

- Index.html
- Main.js
- Preload.js

Ejercicio 1

Vamos a crear la funcionalidad del cronómetro básica.

Crea un evento que inicie la cuenta del cronómetro mediante un intervalo. Como suelen utilizarse las décimas de segundo para estos casos, el intervalo en vez de cada segundo debería ser cada décima de segundo (100 en vez de 1000 en el intervalo, recuerda que se mide en milisegundos).

Ten en cuenta que debes hacer que cada segundo debe sumar 1 minuto y 60 minutos una hora.

Usa la variable 'intervalo_cronometro' proporcionada en el __main__() para almacenar el intervalo.

Pista: Necesitarás almacenar el valor del temporizador entre intervalos al perder el valor del contador cada vez que se ejecuta.

Ejercicio 2

Crea el método que permita parar el cronómetro. Puedes utilizar el botón de 'Start' ya creado o crear un botón nuevo.

Ejercicio 3

Crea el método que permita formar segmentos en el cronómetro. Esto es que cada vez que apretemos el botón flag, el cronómetro no se parará, pero si se guardará el tiempo en el que se pulsó dicho botón.

Si te fijas en el index.html hay una lista desordenada con id 'tiempos_parciales'. Añade a esa lista los valores parciales.

Pista: document.createElement('li')...

Ejercicio 4

El temporizador necesita algunas funciones. Vamos a hacer la función que establezca el tiempo que va a durar. Antes de crear dicha función, primero

crea el evento que permita recoger los valores de horas:minutos:segundos de los respectivos selectores y muéstralos por consola cuando se pulse 'start'.

Ejercicio 5

Ahora pudiendo recoger esos valores, el evento de 'start' debe poner dichos valores en el html 'text_temporizador'.

Ejercicio 6

Una vez muestras los valores, solo queda añadir la cuenta atrás. Crea un temporizador con 'setTimeout' con la variable 'temporizador_temporizador'. Como solo acepta valores en milisegundos, vamos a tener que hacer conversión de valores. La fórmula matemática sería:

$$ms = (h \times 3600) + (m \times 60) + (s \times 1000)$$

Siendo:

- ms= Los milisegundos obtenidos.
- h= Las horas marcadas.
- m= Los minutos marcados.
- s= Los segundos marcados.

El método que debe activarse con setTimeout (recuerda que al contrario que los intervalos, el método se lanza al final de la cuenta atrás) debe

mostrar una alerta por pantalla indicando que se acabó el tiempo o algún mensaje similar.

El problema es que añadir el setTimeout no va a hacer que nuestro temporizador se actualice en tiempo real. En el ejercicio 7 se especifica que se necesita

Ejercicio 7

Una vez tenemos nuestro temporizador, debemos añadir un intervalo con setInterval que actualice la cuenta atrás. Para ello, en el mismo evento que comienza el temporizador, debes de comenzar un intervalo.

Crea la variable 'intervalo_temporizador' tal como se crearon las otras variables globales.

Utilizando dicha variable, cuando se activa pulsa el botón 'start' crea el intervalo que haga la cuenta atrás. Necesitarás 3 variables para horas, minutos y segundos.

Ejercicio 8

Crea el método que reinicie el temporizador. Los valores deben volver a 0 o por defecto.

Extra

Ahora que nos hemos peleado bastante con intervalos, os invito a mejorar las funcionalidades del reloj para practicar:

- Modifica visualmente la página para dejarla con una estética más agradable.
- Crea el método que permita pausar el temporizador. Recuerda que debes limpiar el temporizador y el intervalo. Al darle a 'start' de nuevo debería empezar desde donde está si los valores no son 0 tanto el intervalo como el temporizador.
- Crea alarmas con setTimeout en el reloj estándar, que muestren un mensaje por pantalla. Recoge la hora especificada y en el intervalo del reloj debe comprobar si alguna de las alarmas coincide. Utiliza alguna estructura de datos de las que vimos (listas, pilas o colas).