UT6-IV-TEMPORIZADORES (TIMERS)

6-IV.1- ¿Qué son los temporizadores?

El control del tiempo es un elemento fundamental de la programación. Es fundamental, por ejemplo, en juegos, en creación de animaciones, en publicidad y en otras muchas áreas.
Desde el lado del cliente, el objeto Window tiene métodos relacionados con el control del tiempo.
Estos métodos permiten ejecutar un determinado código tras haber transcurrido un tiempo, el cuál es configurable. Para poder hacer esta tarea, estos métodos necesitan de una función callback.
El hecho de que se ejecute un código cuando ocurra un tiempo concreto nos aproxima a la gestión de eventos, que es la base de uno de los siguientes temas.

6-IV.2- WindowOrWorkerGlobalScope.setTimeout

https://developer.mozilla.org/es/docs/Web/API/setTimeout
El método ***setTimeout()*** del mixin WindowOrWorkerGlobalScope establece un temporizador que ejecuta una función o una porción de código después de que transcurre un tiempo establecido.

Sintaxis

```
var idTemporizador = scope.setTimeout(funcion[, retraso, parametro1, parametro2, ...]);
var idTimeout = scope.setTimeout(funcion[, retraso]);
var idTimeout = scope.setTimeout(codigo[, retraso]);
```

Parámetros

función: Una function para ejecutar después de que expire el temporizador.
codigo: Una sintaxis opcional que le permite incluir una cadena ***en lugar de*** una función, la cual es compilada y ejecutada cuando el temporizador expira. Esta sintaxis no se recomienda por las mismas razones que hacen del uso de eval() un **riesgo de seguridad**.
retraso **(Optional)**: Tiempo, en milisegundos (milésimas de segundo), que el temporizador debe esperar antes de ejecutar la función o el código. Si se omite este parámetro se usa el valor 0. Tenga en cuenta que ***el retraso real puede ser más prolongado***; ver más abajo Reasons for delays longer than specified.
param1, ..., paramN (Optional): Parámetros adicionales que se pasan a la función especificada por ***func*** una vez el temporizador expira.

Nota: (consultar la página en caso de uso de IE-9)

El valor retornado **IDtemporizador** es numérico y no es cero; **identifica** el temporizador creado con la llamada a setTimeout(); este valor puede pasarse a WindowOrWorkerGlobalScope.**clearTimeout()** para **cancelar el temporizador**.

Puede ser útil advertir que setTimeout() y setInterval() ***comparten la misma pila de IDs***, y que tanto clearTimeout() como clearInterval() pueden intercambiarse. Por claridad, sin embargo, debe hacerlos coincidir para evitar confusiones cuando mantenga su código.

Ejemplo-0:

El siguiente ejemplo establece dos botones simples en una página web y los engancha a las rutinas setTimeout() y clearTimeout(). Presionando el primer botón establecerá un temporizador que llama un diálogo de alerta después de tres segundos y guarda el id del temporizador para usarlo con clearTimeout(). Opcionalmente puede cancelar este temporizador presionando el segundo botón.

6-IV.3- WindowOrWorkerGlobalScope.setInterval()

https://developer.mozilla.org/es/docs/Web/API/setInterval

Ejecuta una función o un fragmento de código **de forma repetitiva**, ***cada vez que termina el periodo de tiempo determinado***. Devuelve un ID de proceso.

Sintaxis

var procesoID = window.setInterval(función, intervaloDeTiempo[, parámetro1, parámetro2, ... , parámetroN]);

var procesoID = window.setInterval(código, intervaloDeTiempo);

Parámetros

función: La function que será ejecutada cada intervaloDeTiempo milisegundos.

código: Una sintaxis opcional permite introducir una cadena en lugar de una función, la cual es evaluada y ejecutada cada intervaloDeTiempo milisegundos. Se recomienda evitar esta sintaxis por la misma razón por la que el comando eval() conlleva problemas de seguridad.

intervaloDeTiempo: El tiempo en milisegundos (1/1000 de segundo, ms) que se debe esperar entre cada ejecución de la función o del código. Si el valor es menor que 10, se usará 10 en su lugar. El tiempo entre cada ejecución puede ser mayor al que indicamos, para mayor información puedes revisar el siguiente artículo: Reasons for delays longer than specified in setTimeout().

El parámetro intervaloDeTiempo es convertido en un entero de 32 bits con signo en el IDL, por lo que el valor más grande que puede tener es 2,147,483,647 milisegundos, aproximadamente 24.8 días.

parámetro1, ..., parámetroN ***(Optional)***: Parámetros adicionales que se pasan a la función a ejecutar.
Nota: (consultar la página en caso de uso de IE-9)

Valor de Retorno

El valor de retorno procesoID es un valor numérico distinto de cero que **identifica al temporizador** que fue creado al llamar setInterval(); ***este valor puede ser usado como parámetro en la función*** Window.**clearInterval()** (en-US) para ***detener el temporizador***.

Las funciones setInterval() y setTimeout() comparten la misma pila de IDs, por lo que, técnicamente, los comandos clearInterval() y clearTimeout() pueden usarse indiscriminadamente. Sin embargo, por motivos de claridad y mantenimiento, es importante usarlos como corresponde.

Nota: El argumento intervaloDeTiempo se convierte aun entero con signo de 32 bits. Esto limita efectivamente al intervaloDeTiempo a valores de 2147483647 ms, ya que se especifica como entero con signo en el IDL.

Ejemplos a probar:

Ej-0-setInterval.
Uso-setInterval1

6-IV.4- window.clearTimeout

https://developer.mozilla.org/es/docs/Web/API/clearTimeout
Borra el retraso asignado por window.setTimeout() (en-US).

Sintaxis

```
window.clearTimeout(timeoutID)
```

timeoutID es el ID del timeout que desee borrar, retornado por window.setTimeout() (en-US).

Ejemplo

Revise cualquiera de los ejemplos vistos.

6-IV.4- WindowTimers.clearInterval()

https://developer.mozilla.org/es/docs/Web/API/clearInterval

Cancela una acción reiterativa que se inició mediante una llamada a setInterval (en-US).

Sintaxis

```
window.clearInterval(intervalID)
```

intervalID es el identificador de la acción reiterativa que se desea cancelar. Este ID se obtiene a partir de setInterval().

Ejemplo

Ver el ejemplo relativo a setInterval().