3.1. Manejo del tiempo en JavaScript.

El objeto **Date** se creó para almacenar fechas y horas.

var = miFecha = new Date ();

Ahora, la variable *miFecha* contendrá la fecha y hora actual. Para mostrar la fecha y la hora actuales en una página web:

```
<div id="laHora"></div>
<script>
var miFecha = new Date();
var texto = document.getElementById('laHora');
texto.innerHTML=miFecha;
</script>
```

Para mostrar la hora, minutos y segundos como normalmente los podemos ver en una página web, debemos usar los siguientes métodos:

- ***getHours()*.** Método que extrae las horas del objeto Date actual.
- ***getMinutes()*.** Método que extrae los minutos del objeto Date actual.
- ***getSeconds()*.** Método que extrae los segundos del objeto Date actual.

- ver ejemplos - *01_objeto_Date.html* (adjunto)

El problema que se puede encontrar es que las funciones anteriores no devuelven dos dígitos siempre.

Para evitar ese "error", debemos tener en cuenta la condición de que cuando el valor sea inferior a 10 (es decir de 0 a 9) nos coloque un '0' delante. De esta forma la hora "8:00", se representaría como "08:00".

- ver ejemplo - *01_objeto_Date_con_cero.html* (adjunto).

Hasta ahora se ha mostrado la hora en formato 24 horas, pero si se quiere representar en formato de 12 horas, añadiendo los prefijos de "pm" y "am" dependiendo de la hora en la

que nos encontremos, deberemos establecer la condición de que si la hora es mayor que 12, a esa hora le restaremos 12 y le añadimos la etiqueta “pm”. De esta forma, si son las 13 horas, obtendríamos que 13 -12 = 1 y con la etiqueta “pm”. En otro caso, es decir, si la hora es menos que 12, no calculamos nada y añadimos la etiqueta “am”.

3.2. Las funciones setTimeout y setInterval.

Para realizar una mejora al código del reloj del apartado anterior, JavaScript cuenta con las siguientes funciones:

- ***setTimeout*** (Función_a_llamar, milisegundos). Ejecutará la función transcurrido el tiempo indicado en el segundo parámetro, en milisegundos (1s = 1000ms).
- ***setInterval*** (Función_a_llamar, milisegundos). Ejecutará la función de manera periódica según los milisegundos introducidos en el segundo parámetro.
- ***clearInterval().*** para la ejecución iniciada con setInterval().

  var elCrono = setInterval(crono, 1000);

  clearInterval(elCrono);
- ***clearTimeout().*** Para la ejecución iniciada con setTimeout().

  var elTemporizador = setTimeout(laFuncion, 5000);

  clearTimeout (elTemporizador);

En el caso del reloj que se está implementando, está claro que la segunda función, setInterval, es la más adecuada para que el reloj se vaya actualizando cada segundo.

- ver ejemplo - *01_objeto_Date_crono.html* (adjunto)

Se usa el evento ***onload*** del objeto ***windows***, lo que implica que en la carga de la página web, se ejecute la sentencia ***setInterval*** programando la ejecución de la función ***crono*** cada segundo.