CREACION DE UN RELOJ

El objeto Date se creó en JavaScript para almacenar fechas y horas. Una vez creado dicho objeto, es posible modificarlo y realizar los cálculos que el programador crea convenientes para modificar las fechas y las horas.

La creación de este objeto se podría hacer de la siguiente forma:

```
var miFecha = new  Date();
```

Ahora, la variable miFecha contendrá la fecha y hora actual. Si se desea mostrar la fecha y la hora actuales en un campo de una página web, es tan fácil como utilizar el siguiente código:

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

El campo laHora, en el momento de la ejecución, podrá tener un valor parecido al siguiente:

Mon Jan 07 2019 17:14:45 GMT+0100 (hora estándar de Europa central)

El problema es que el aspecto de la hora que se muestra no es el que normalmente se visualiza en una página web y, por lo tanto, hay que utilizar algunas funciones (métodos) para mostrar la hora, los minutos y los segundos como son:

- *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.

El siguiente ejemplo sí mostrará la hora en un formato más convencional:

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

El resultado puede ser parecido al siguiente:

18:46:58

El problema que se puede encontrar es que las funciones anteriores (getHours, getMinutes y getSeconds) no devuelven dos dígitos siempre. Si, por ejemplo, la función getSeconds tiene que devolver tres segundos, devolverá el número 3 y no 03, con lo cual habrá que modificar o mejorar el código anterior:

```
<div id="laHora"></div>
<script>
    var miFecha = new Date();
    var horas = miFecha.getHours();
    var minutos = miFecha.getMinutes();
    var segundos = miFecha.getSeconds();
    if (horas<10){horas='0'+horas;}
    if (minutos<10){minutos='0'+minutos;}
    if (segundos<10){segundos='0'+segundos;}

    var texto = document.getElementById('laHora');
    texto.innerHTML=horas+':'+minutos+':'+segundos;
</script>
```

En este momento, el reloj implementado mostrará siempre la hora de forma más homogénea.

Imagínese que se quiere mostrar la hora en formato 12 horas en vez de en el de 24, como se estaba haciendo. Habrá que modificar el código de la siguiente manera:

```
<div id="laHora"></div>
<script>
    var miFecha = new Date();
    var horas = miFecha.getHours();
    var minutos = miFecha.getMinutes();
    var segundos = miFecha.getSeconds();
    if (horas > 12){
        ampm='pm';
        horas-=12;
    }else{
        ampm='am';
    }
    if (horas<10){horas='0'+horas;}
    if (minutos<10){minutos='0'+minutos;}
    if (segundos<10){segundos='0'+segundos;}

    var texto = document.getElementById('laHora');
    texto.innerHTML=horas+':'+minutos+':'+segundos+' '+ampm;
</script>
```

Como se puede observar, lo que se hace es comprobar si son más de las 12 o no y se ajustan las horas colocando el sufijo correspondiente, ya sea *am* o *pm*.

La hora ahora podrá mostrarse en un formato parecido al siguiente:

Para realizar una mejora sustancial al ejercicio desarrollado anteriormente, JavaScript ofrece las siguientes funciones:

- *setTimeout (Función_a_llamar,milisegundos).* Ejecutará la función Función_a_llamar transcurrido el tiempo indicado en el segundo parámetro. El tiempo se expresa en milisegundos. Es importante introducir en el primer parámetro solamente el nombre de la función sin paréntesis.
- *setInterval (Función_a_llamar,milisegundos).* Función parecida a la anterior, pero, en este caso, se ejecutará la función Función_a_llamar de manera periódica según los milisegundos introducidos en el segundo parámetro.
- *clearInterval().* Para la ejecución iniciada con setInterval(). Véase un ejemplo de uso:

```
var elCrono = setInterval(crono, 1000);
clearInterval(elCrono);
```

- *clearTimeout().* Para la ejecución iniciada con setTimeout(). Véase un ejemplo de uso:

```
var elTemporizador = setTimeout(laFuncion, 5000);
clearTimeout(elTemporizador);
```

En el caso del reloj que se está implementando, es obvio que la segunda función será la más indicada para actualizar el cronómetro cada segundo. El código completo de la página web quedará de la siguiente manera:

```
<!DOCTYPE html>
<html>
<head>
<title>Reloj</title>
</head>
<body>
<div id="laHora"></div>
<script>
function crono(){
var elCrono;
var miFecha = new Date();
var horas = miFecha.getHours();
var minutos = miFecha.getMinutes();
var segundos = miFecha.getSeconds();
if (horas > 12){
   ampm='pm';
   horas-=12;
}else{
   ampm='am';
}
if (horas<10){horas='0'+horas;}
if (minutos<10){minutos='0'+minutos;}
if (segundos<10){segundos='0'+segundos;}

var texto = document.getElementById('laHora');
texto.innerHTML=horas+':'+minutos+':'+segundos+' '+ampm;
}
window.onload=function(){
   elCrono = setInterval(crono,1000);
}
</script>
</body>
</html>
```