8.1. Relojes, contadores e intervalos de tiempo

En ocasiones, algunas páginas web muestran un reloj con la hora actual. Si el reloj debe actualizarse cada segundo, no se puede mostrar la hora directamente en la página HTML generada por el servidor. En este caso, aunque existen alternativas realizadas con Java y con Flash, la forma más sencilla de hacerlo es mostrar la hora del ordenador del usuario mediante JavaScript.

Para crear y mostrar un reloj con JavaScript, se debe utilizar el objeto interno `Date()` para crear fechas/horas y las utilidades que permiten definir contadores, para actualizar el reloj cada segundo.

El objeto `Date()` es una utilidad que proporciona JavaScript para crear fechas y horas. Una vez creado un objeto de tipo fecha, es posible manipularlo para obtener información o realizar cálculos con las fechas. Para obtener la fecha y hora actuales, solamente es necesario crear un objeto `Date()` sin pasar ningún parámetro:

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

Utilizando el código anterior, se puede construir un reloj muy básico que no actualiza su contenido:

```
var fechaHora = new Date();
document.getElementById("reloj").innerHTML = fechaHora;

<div id="reloj" />
```

Cuando se carga la página, el ejemplo anterior mostraría un texto parecido al siguiente en el `<div>` reservado para el reloj:

```
Mon May 04 2009 13:36:10 GMT+0200 (Hora de verano romance)
```

Este primer reloj construido presenta muchas diferencias respecto al reloj que se quiere construir. En primer lugar, muestra más información de la necesaria. Además, su valor no se actualiza cada segundo, por lo que no es un reloj muy práctico.

El objeto `Date()` proporciona unas funciones muy útiles para obtener información sobre la fecha y la hora. Concretamente, existen funciones que devuelven la hora, los minutos y los segundos:

```
var fechaHora = new Date();
var horas = fechaHora.getHours();
var minutos = fechaHora.getMinutes();
var segundos = fechaHora.getSeconds();

document.getElementById("reloj").innerHTML = horas+':'+minutos+':'+segundos;

<div id="reloj" />
```

Utilizando las funciones `getHours()`, `getMinutes()` y `getSeconds()` del objeto `Date`, el reloj solamente muestra la información de la hora. El resultado del ejemplo anterior sería un reloj como el siguiente:

```
20:9:21
```

Si la hora, minuto o segundo son menores que `10`, JavaScript no añade el `0` por delante, por lo que el resultado no es del todo satisfactorio. El siguiente código soluciona este problema añadiendo un `0` cuando sea necesario:

```
var fechaHora = new Date();
var horas = fechaHora.getHours();
var minutos = fechaHora.getMinutes();
var segundos = fechaHora.getSeconds();

if(horas < 10) { horas = '0' + horas; }
if(minutos < 10) { minutos = '0' + minutos; }
if(segundos < 10) { segundos = '0' + segundos; }

document.getElementById("reloj").innerHTML = horas+':'+minutos+':'+segundos;

<div id="reloj" />
```

Ahora el reloj muestra correctamente la hora:

```
20:14:03
```

Si se quiere mostrar el reloj con un formato de 12 horas en vez de 24, se puede utilizar el siguiente código:

```
var fechaHora = new Date();
var horas = fechaHora.getHours();
var minutos = fechaHora.getMinutes();
var segundos = fechaHora.getSeconds();

var sufijo = ' am';
if(horas > 12) {
  horas = horas - 12;
  sufijo = ' pm';
}

if(horas < 10) { horas = '0' + horas; }
if(minutos < 10) { minutos = '0' + minutos; }
if(segundos < 10) { segundos = '0' + segundos; }

document.getElementById("reloj").innerHTML = horas+':'+minutos+':'+segundos+sufijo;

<div id="reloj" />
```

Utilizando el código anterior, el reloj ya no muestra la hora como `20:14:03`, sino que la muestra en formato de 12 horas y con el sufijo adecuado:

```
08:14:03 pm
```

Para completar el reloj, sólo falta que se actualice su valor cada segundo. Para conseguirlo, se deben utilizar unas funciones especiales de JavaScript que permiten ejecutar determinadas instrucciones cuando ha transcurrido un determinado espacio de tiempo.

La función `setTimeout()` permite ejecutar una función una vez que haya transcurrido un periodo de tiempo indicado. La definición de la función es:

```
setTimeout(nombreFuncion, milisegundos);
```

La función que se va a ejecutar se debe indicar mediante su nombre sin paréntesis y el tiempo que debe transcurrir hasta que se ejecute se indica en milisegundos. De esta forma, si se crea una función encargada de mostrar la hora del reloj y se denomina `muestraReloj()`, se puede indicar que se ejecute dentro de 1 segundo mediante el siguiente código:

```
function muestraReloj() {
  var fechaHora = new Date();
  var horas = fechaHora.getHours();
  var minutos = fechaHora.getMinutes();
  var segundos = fechaHora.getSeconds();

  if(horas < 10) { horas = '0' + horas; }
  if(minutos < 10) { minutos = '0' + minutos; }
  if(segundos < 10) { segundos = '0' + segundos; }

  document.getElementById("reloj").innerHTML = horas+':'+minutos+':'+segundos;
}

setTimeout(muestraReloj, 1000);

<div id="reloj" />
```

No obstante, el código anterior simplemente muestra el contenido del reloj 1 segundo después de que se cargue la página, por lo que no es muy útil. Para ejecutar una función de forma periódica, se utiliza una función de JavaScript muy similar a `setTimeout()` que se denomina `setInterval()`. Su definición es:

```
setInterval(nombreFuncion, milisegundos);
```

La definición de esta función es idéntica a la función `setTimeout()`, salvo que en este caso, la función programada se ejecuta infinitas veces de forma periódica con un lapso de tiempo entre ejecuciones de tantos milisegundos como se hayan establecido.

Así, para construir el reloj completo, se establece una ejecución periódica de la función `muestraReloj()` cada segundo:

```
function muestraReloj() {
  var fechaHora = new Date();
  var horas = fechaHora.getHours();
  var minutos = fechaHora.getMinutes();
  var segundos = fechaHora.getSeconds();

  if(horas < 10) { horas = '0' + horas; }
  if(minutos < 10) { minutos = '0' + minutos; }
  if(segundos < 10) { segundos = '0' + segundos; }

  document.getElementById("reloj").innerHTML = horas+':'+minutos+':'+segundos;
}

window.onload = function() {
  setInterval(muestraReloj, 1000);
}

<div id="reloj" />
```

Empleando el objeto `Date` y sus funciones, es posible construir *"cuentras atrás"*, es decir, relojes que muestran el tiempo que falta hasta que se produzca un evento. Además, las funciones `setTimeout()` y `setInterval()` pueden resultar muy útiles en otras técnicas de programación.

← **Anterior** **Siguiente →**

Indice de contenidos