



# **JAVASCRIPT**

## **Capítulo 8**

*Lenguajes de Marcas y Sistemas de Gestión de Información*

*Curso 2019/2020*



# Capítulo 8: OTRAS UTILIDADES

1. Introducción
2. Relojes, Contadores e Intervalos de Tiempo
3. Calendario
4. *Tooltip*
5. Galería de Imágenes



# 1. INTRODUCCIÓN

Existen muchas otras utilidades que es necesario conocer para desarrollar aplicaciones completas. A continuación se muestran algunas de las utilidades básicas más comunes:

- Relojes.
- Contadores.
- Intervalos de Tiempo.
- Calendario.
- *Tooltip*.
- Galería de Imágenes.



## 2. RELOJES, CONTADORES E INTERVALOS DE TIEMPO

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. La forma más sencilla de hacerlo es mediante JavaScript.

Se debe utilizar el objeto `Date()` para crear fechas/horas y las utilidades que permiten definir **contadores**, para actualizar el reloj cada segundo.

Una vez creado un objeto de tipo fecha, es posible **manipularlo** para obtener información o realizar cálculos con las fechas:

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



## 2. RELOJES, CONTADORES E INTERVALOS DE TIEMPO

Se puede construir un reloj muy básico:

```
var fechaHora = new Date();  
  
document.getElementById("reloj").innerHTML = fechaHora;  
  
<div id="reloj" />
```

Cuando se carga la página, se mostrará un texto parecido al siguiente:

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

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



## 2. RELOJES, CONTADORES E INTERVALOS DE TIEMPO

El objeto `Date()` proporciona funciones muy útiles para obtener información sobre la fecha y la hora:

```
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" />
```

El reloj solamente muestra la información de la hora:

20:9:21



## 2. RELOJES, CONTADORES E INTERVALOS DE TIEMPO

Si la hora, minuto o segundo son menores que 10, JS no añade el 0 por delante. Este problema se soluciona 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" />
```

**Resultado**

20:14:03



## 2. RELOJES, CONTADORES E INTERVALOS DE TIEMPO

Sólo falta que se actualice su valor cada segundo. Se deben utilizar unas funciones especiales de JS 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 (en milisegundos).

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**.



## + 2. RELOJES, CONTADORES E INTERVALOS DE TIEMPO

9

```
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" />
```

**¡¡ El código muestra el contenido del reloj 1 segundo después de que se cargue la página !!**



## 2. RELOJES, CONTADORES E INTERVALOS DE TIEMPO

Para ejecutar una función de forma periódica, se utiliza una función de JS muy similar a `setTimeout()` que se denomina `setInterval()`.

Su definición es:

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

La definición 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.

## + 2. RELOJES, CONTADORES

# INTERVALOS DE TIEMPO

E

11

```
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" />
```



## 3. CALENDARIO

Existen **muchas formas diferentes** de indicar fechas:

dd/mm/aa, dd/mm/aaaa, aaaa/mm/dd, dd-mm-aa, dd mm aaaa, etc.

Los métodos más utilizados suelen ser un cuadro de texto en el que tiene que insertar la fecha completa (**indicándole el formato que debe seguir**) o **listas desplegables** para los días, meses y años.

Para el usuario suele ser más cómodo que **la aplicación incluya un calendario** en el que pueda indicar la fecha pinchando sobre el día elegido.

## 3. CALENDARIO

?	Agosto, 2007						×
«	<	Hoy				>	»
sem	Lun	Mar	Mié	Jue	Vie	Sáb	Dom
31			1	2	3	4	5
32	6	7	8	9	10	11	12
33	13	14	15	16	17	18	19
34	20	21	22	23	24	25	26
35	27	28	29	30	31		
Seleccionar fecha							

Este método es **menos propenso a cometer errores**, ya que si el calendario está bien construido, no es posible introducir fechas inválidas y tampoco es posible insertar las fechas en un formato incorrecto.

## 3. CALENDARIO

Realizar un calendario completo en JS no es sencillo. Afortunadamente, existen **scripts gratuitos** para mostrar calendarios y que permiten su uso libre incluso en aplicaciones comerciales.

Uno de los más completos es el desarrollado por **DynArch**.

Se puede descargar **gratuitamente**. El archivo descargado incluye todos los scripts necesarios, su documentación, ejemplos de uso, diferentes estilos CSS para el calendario, etc...



## 3. CALENDARIO: PASO 1

Enlazar los archivos JavaScript y CSS requeridos:

```
<head>
```

```
...
```

```
<style type="text/css">
```

```
@import url("css/calendar-estilo.css");
```

```
</style>
```

```
<script type="text/javascript" src="js/calendar.js" />
```

```
<script type="text/javascript" src="js/calendar-es.js" />
```

```
<script type="text/javascript" src="js/calendar-setup.js" />
```

```
...
```

```
</head>
```

## 3. CALENDARIO: PASO 2

Añadir el código necesario para el elemento que va a mostrar el calendario:

```
<p>Introduce la fecha pulsando sobre la imagen del  
calendario</p>
```

```
<input type="text" name="date" id="fecha"  
readonly="readonly" />
```

```

```

- Cuadro de texto llamado `fecha` que almacena el valor introducido por el usuario. Como se le ha asignado un atributo `readonly="readonly"`, el usuario no puede modificar su valor.
- Imagen que se utiliza para activar el calendario.





## 3. CALENDARIO: PASO 3

Configurar el calendario:

```
<script type="text/javascript">

window.onload = function() {

    Calendar.setup({

        inputField: "fecha",

        ifFormat:   "%d / %m / %Y",

        button:     "selector"

    });

}

</script>
```

Una vez enlazados los archivos del calendario y preparado el código XHTML, es necesario **inicializar y configurar** el calendario. La configuración del calendario consiste en establecer el valor de alguna de sus propiedades.



## 3. CALENDARIO: PASO 3

Las **propiedades básicas** imprescindibles son:

- **inputField**: atributo `id` del elemento en el que se mostrará la fecha seleccionada, en este ejemplo sería `fecha`.
- **ifFormat**: formato en el que se mostrará la fecha una vez seleccionada.
- **button**: atributo `id` del elemento que se pulsa (botón, imagen, enlace) para mostrar el calendario de selección de fecha. En este ejemplo, el `id` de la imagen es `selector`.

## 4. *TOOLTIP*

Son **pequeños recuadros** de información que aparecen al posicionar el ratón sobre un elemento. Se utilizan para ofrecer **información adicional** sobre el elemento seleccionado o para mostrar al usuario pequeños **mensajes de ayuda**.

Realizar un *tooltip* completo mediante JS es una **tarea compleja**, sobre todo por la dificultad de mostrar el mensaje correctamente en función de la posición del ratón. Afortunadamente, **existen scripts** que ya están preparados para generar cualquier tipo de *tooltip*.

La librería que se va a utilizar se denomina **overLIB**.



## 4. *TOOLTIP*: PASO 1

Enlazar los archivos JavaScript requeridos:

```
<script type="text/javascript" src="js/overlib.jsoverLIB (c) Erik Bosrup --></script>
```

Se guarda el archivo JavaScript en el sitio adecuado y se enlaza directamente desde la cabecera de la página HTML.

Los *tooltips* sólo requieren que se enlace un único archivo JavaScript (`overlib.js`). El comentario que incluye el código XHTML es el **aviso de copyright** de la librería, que es **obligatorio** incluirlo para poder usarla.




## 4. *TOOLTIP*: PASO 2

Definir el texto que activa el *tooltip* y su contenido:

```
<p>Lorem ipsum dolor sit amet, <a href="#" onmouseover="return  
overlib('Prueba de un tooltip básico y muy sencillo.');"   
onmouseout="return nd();">consectetuer adipiscing elit</a>.  
Etiam eget metus. Proin varius auctor tortor. Cras augue  
neque, porta vitae, vestibulum nec, pulvinar id, nibh. Fusce  
in arcu. Duis vehicula nonummy orci.</p>
```

Se incluyen como enlaces de tipo `<a>` para los que se definen los eventos `onmouseover` y `onmouseout`. Cuando el ratón se pasa por encima del enlace anterior, se muestra el *tooltip* con el aspecto por defecto:


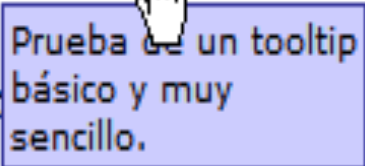
met, consectetuer adipiscing elit. Etiam eget metus. Proin vari  
el adipiscing tempor, n  Prueba de un tooltip básico y muy  
sencillo. endu



## 4. *TOOLTIP*: PASO 2

La librería **overLIB** permite configurar completamente el aspecto y comportamiento de cada *tooltip*.

```
<a href="#" onmouseover="return overlib('Prueba de un
tooltip básico y muy sencillo.', CENTER, WIDTH, 120);"
onmouseout="return nd();">consectetur adipiscing
elit</a>
```

dolor sit amet, consectetur adipiscing elit. Etiam eget metus  
 isis, lacus vel adipisc   quam tincidunt ante, s  
 potenti.



## 5. GALERÍAS DE IMÁGENES (LIGHTBOX)

Muchos sitios web utilizan **galerías de imágenes** para mostrar sus productos y servicios. Este tipo de galerías muestran una serie de **miniaturas** de imágenes que se amplían al pinchar sobre cada imagen.

El uso de técnicas basadas en JS ha supuesto una revolución en la creación de las galerías de imágenes y ha permitido mejorar la experiencia de navegación del usuario.

La técnica se conoce como **Lightbox** y funciona correctamente en todos los navegadores y permite mantener la semántica del documento (no ensucia la página con código JavaScript).

## + 5. GALERÍAS DE IMÁGENES (LIGHTBOX)

Se enlazan los archivos JavaScript y CSS requeridos. La página XHTML original contiene un enlace simple que abre una nueva página en la que se puede ver ampliada la imagen original:

```
<a href="images/image.jpg" title="Título de la
imagen 1">Imagen 1</a>
```

Para **activar Lightbox**, tan solo es necesario incluir el siguiente atributo rel:

```
<a href="images/image-1.jpg" rel="lightbox"
title="Título de la imagen 1">Imagen 1</a>
```



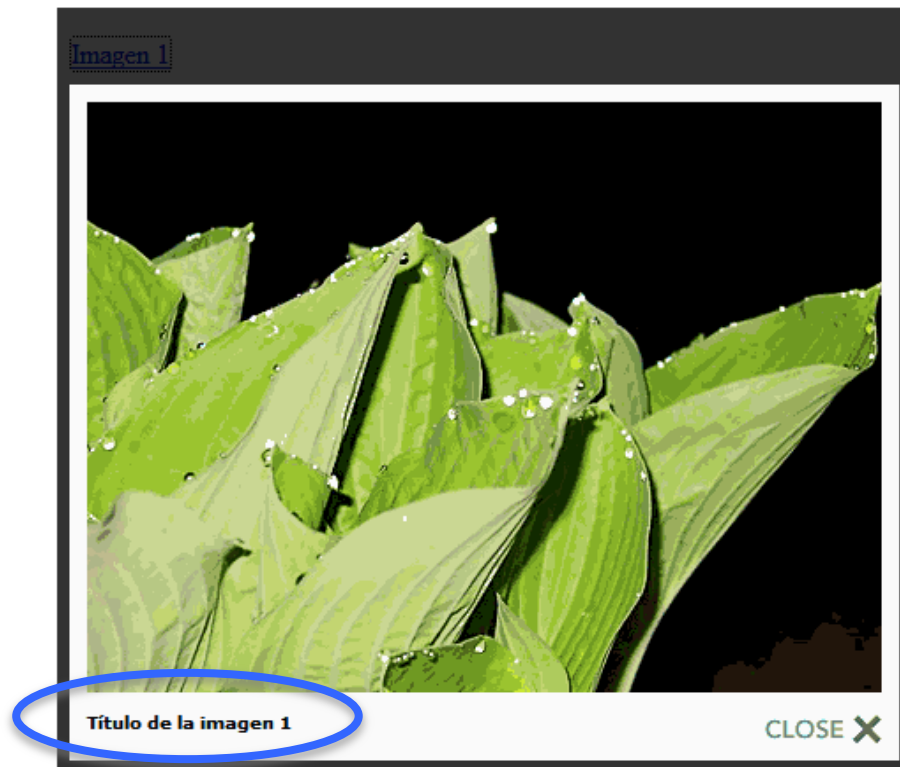
## + 5. GALERÍAS DE IMÁGENES (LIGHTBOX)

25

[Imagen 1](#)

Al pinchar en la imagen, se mostrará la imagen ampliada centrada en la misma página:

**Lightbox** muestra como título de la imagen el valor del atributo `title` del enlace

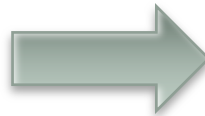


## + 5. GALERÍAS DE IMÁGENES (LIGHTBOX)

Normalmente, Lightbox no se utiliza con enlaces de texto sino con imágenes que contienen enlaces:

```
<a href="images/image-1.jpg" rel="lightbox" title="Título de la imagen 1"></a>
```

Ahora al pinchar sobre la imagen pequeña:



<https://www.lokeshdhakar.com/projects/lightbox2/>