APRENDERAPROGRAMAR.COM

DATE JAVASCRIPT. DATE.NOW, DATE.PARSE, DATE.UTC. DIFERENCIAS ENTRE GMT Y UTC Ó LOCAL. GETMONTH, GETDATE, GETDAY. (CU01162E)

Sección: Cursos
Categoría: Tutorial básico del programador web: JavaScript desde cero

Fecha revisión: 2029

Resumen: Entrega nº62 del Tutorial básico "JavaScript desde cero".

Autor: César Krall

OBJETO DATE JAVASCRIPT

Con frecuencia cuando creamos webs o apps tenemos que trabajar con fechas y calendarios. Por ejemplo en la página web de un hotel o un restaurante es posible que trabajemos con fechas y horas de comienzo de reserva, de fin de reserva, etc. JavaScript dispone del objeto predefinido Date para facilitar el trabajo con fechas.

El manejo del "tiempo" es un aspecto controvertido en la programación. La primera dificultad y más obvia es que existen cientos de dispositivos electrónicos y no todos manejan la misma fecha. Por ejemplo mi computador personal puede indicar que hoy es lunes 25 de enero de 2048 y la hora es las 12:35:44 mientras que el servidor al que estoy conectado, suponiendo que está en mi misma ciudad, puede indicar que hoy es lunes 25 de enero de 2048 y la hora es las 12:33:21. Otro servidor, en otro país, puede indicar que la fecha es domingo 24 de enero de 2048 y la hora es las 23:51:15 debido a la diferencia horaria entre países. Aquí, claro está, nos estaríamos refiriendo a horas locales. Para poder disponer de horas de referencia globales para todo el mundo se crearon estándares como:

a) GMT o tiempo medio de Greenwich: ha sido un estándar ampliamente usado. GMT era el tiempo medido en el observatorio británico de Greenwich, que se definió como meridiano cero de la tierra. Así cada país podía expresar su hora en función de la hora del meridiano de Greenwich. Por ejemplo GMT+0 indicaba que la hora era la misma que la hora oficial en el meridiano de Greenwich, mientras que GMT+3 indicaba que la hora eran 3 horas más que en el meridiano de Greenwich (es decir, si es Greenwich eran las 09:00 en un país cuya hora fuera GMT+3 serían las 12:00).

b) UTC o tiempo universal coordinado: es el estándar que se ha adoptado como referencia que indica un tiempo único independientemente de en qué lugar del planeta nos encontremos, basado en las mediciones de relojes atómicos distribuidos por distintos países. Es el estándar que se está imponiendo en los sistemas informáticos.

La circunferencia terrestre se dividió en 24 husos horarios de modo que cada huso quedaba referenciado a un tiempo común (lo que se denominaba el tiempo de Greenwich).

Aunque desde el punto de vista de su definición técnica UTC y GMT no son lo mismo, a efectos prácticos hablar de GMT+3 es lo mismo que hablar de UTC+3.

A pesar de este gran avance, el trabajo con el tiempo sigue presentando grandes dificultades en los sistemas informáticos, tanto por la falta de sincronización entre dispositivos como por la disparidad en cuanto a cómo medir el tiempo y con qué grado de precisión, existencia de horarios de verano e invierno, etc. Por ello encontrarás que es relativamente frecuente encontrar que a medida que los lenguajes de programación evolucionan vayan introduciendo cambios en la forma de manejar el tiempo.

Una cuestión a tener en cuenta es que el tiempo oficial de Greenwich, tiempo GMT ó UTC, no se corresponde con el tiempo local el Greenwich debido a la existencia del horario de verano. Por ejemplo, Lisboa se encuentra en el huso horario de Greenwich, pero en verano la hora local está adelantada una hora respecto al tiempo oficial de Greenwich por motivos de ahorro energético. Esto da lugar a que el 5 de agosto a las 21:00 en Lisboa se corresponda con el 5 de agosto 20:00 UTC debido al horario de verano, a pesar de que Lisboa se encuentre en el mismo huso horaria que Greenwich.

OBJETO DATE JAVASCRIPT

JavaScript nos provee de un objeto predefinido (existente) del lenguaje que podemos usar para trabajar con fechas en nuestros scripts: el objeto Date. Este es uno de los objetos predefinidos de JavaScript, ya que existen otros (como Math, RegExp, etc.). Además de un objeto predefinido Date define un tipo de dato en JavaScript.

MÉTODOS DEL OBJETO PREDEFINIDO DATE

El objeto Date (predefinido) nos proporciona estos métodos:

MÉTODO	UTILIDAD	EJEMPLOS aprenderaprogramar.com
Date.now()	Devuelve el número de milisegundos transcurridos desde el 1 de enero de 1970 a las 00:00:00 horas UTC hasta el instante actual UTC según el tiempo del sistema.	alert('Milisegundos desde 1 de enero de 1970: '+Date.now());
Date.parse(fecha)	Transforma el String fecha en una fecha y devuelve el número de milisegundos transcurridos desde el 1 de enero de 1970 a las 00:00:00 UTC hasta el instante actual UTC según el tiempo del sistema.	alert('Milisegundos desde 1 de enero de 1970: '+Date.parse('1970-01-02T00:00:00+0000')); //Devuelve 86400000
Date.UTC(año,mes, dia, hora, min, seg, ms)	Año y mes son obligatorios y el resto opcionales. El año ha de ser mayor que 1900. Los meses van de 0 a 11 y los días de 1 a 31. Devuelve el número de milisegundos transcurridos desde el 1 de enero de 1970 a las 00:00:00 UTC hasta el instante UTC indicado en los argumentos.	alert('Milisegundos desde 1 de enero de 1970 son: '+Date.UTC(1970, 0, 2, 0, 0, 0)); //Devuelve 86400000

El método toString() aplicado a estos métodos devuelve un valor numérico (milisegundos).

CONSTRUCTORES DE OBJETOS TIPO DATE

Además del objeto predefinidio Date, podemos crear objetos Date, que representan "un instante" en el tiempo. Hay cuatro formas distintas de crear objetos Date (constructores):

```
new Date();
new Date(valorEntero);
new Date(fechaString);
new Date(año, mes, día, hora, minuto, segundo, milisegundo);
```

El constructor sin parámetros crea un objeto Date donde el instante de tiempo que representa es el tiempo del sistema en el momento de la creación. El tiempo del sistema es el tiempo de acuerdo con nuestro computador (sistema operativo), y no necesariamente coincidirá con el de otro computador.

Si se invoca Date() sin la palabra new, se obtiene la representación del tiempo actual del sistema. No hay que confundir esta invocación (que sería a una función que devuelve un valor de tipo Date correspondiente al tiempo actual y que cambiará en cada invocación) con la invocación new Date(), que crea un objeto Date que representa el tiempo actual, que queda almacenado en el objeto.

El constructor que recibe un valor entero crea un objeto Date cuyo instante de tiempo está definido por los milisegundos que hayan pasado desde el origen de tiempo (el origen de tiempo en JavaScript está fijado el 1 de enero de 1970 a las 00:00:00 horas, medianoche). Dado que un día tiene 24 horas, una hora 60 minutos, un minuto 60 segundos y un segundo 1000 milisegundos, para representar el 2 de enero de 1970 usaríamos new Date(86400000); , donde el entero entre paréntesis son los milisegundos que contiene un día.

Ejecuta este código y comprueba sus resultados:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head><title>Ejemplo aprenderaprogramar.com</title><meta charset="utf-8">
<script type="text/javascript">
function ejemplo() {
var capturaTiempo = new Date();
alert('capturaTiempo es ' + capturaTiempo);
var unaFecha = new Date(86400000);
alert('El valor numérico 86400000 representa la fecha '+unaFecha);
alert('Date tiempo actual es ' +Date() + ' mientras que capturaTiempo sigue valiendo'+capturaTiempo+
' y unaFecha sigue valiendo '+unaFecha);
}
</script></head>
<body><div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3></div>
<div style="color:blue;" id ="pulsador" onclick="ejemplo()"> Probar </div>
</body></html>
```

El resultado esperado (ten en cuenta que puede diferir según el navegador y sistema operativo que estés usando) es que se muestre:

capturaTiempo es Mon Aug 11 2094 17:22:48 GMT+0100 >> El valor numérico 86400000 representa la fecha Fri Jan 02 1970 00:00:00 GMT+0000 (Hora de verano GMT) >> Date tiempo actual es Mon Aug 11 2094 17:23:31 GMT+0100 mientras que capturaTiempo sigue valiendoMon Aug 11 2094 17:22:48 GMT+0100 y unaFecha sigue valiendo Fri Jan 02 1970 00:00:00 GMT+0000 (Hora de verano GMT)

Comprobamos que el método toString() aplicado a objetos de tipo Date devuelve una representación como cadena de texto de la fecha.

El constructor que recibe un String crea un objeto Date a partir de una cadena de texto que representa una fecha construida según las formas normalizadas válidas. Hay múltiples formas normalizadas (europeas, norteamericanas) y algunos navegadores reconocen más formas que otros, por lo que no vamos a estudiar todas ellas, sino a mostrar ejemplos en una forma recomendada consistente en indicar año-mes-diaThh:mm+xxyy donde +xxyy indica una diferencia horaria respecto a la hora UTC, que puede ser negativa (hora local retrasada respecto a la UTC) o positiva (hora local adelantada respecto a la UTC).

Ejemplo	Comentarios aprenderaprogramar.com
new Date ('2050-12-25T00:00:00+0000') **ó new Date ('2050-12-25')**	25 de diciembre de 2050 a las 00:00:00 horas locales, estando en una zona horaria sin diferencia respecto a la hora UTC (Greenwich)
new Date ('2050-12-25T00:00+0300')	25 de diciembre a las 00:00:00 horas locales estando en una zona horaria con 3 horas de adelanto respecto a la hora UTC, por tanto representa la fecha 24 de diciembre de 2050 a las 21:00:00 horas UTC
new Date ('2050-12-25T12:00-0600')	25 de diciembre a las 00:00:00 horas locales estando en una zona horaria con 6 horas de retraso respecto a la hora UTC, por tanto representa la fecha 24 de diciembre de 2050 a las 18:00:00 horas UTC

El constructor new Date(año, mes, día, hora, minuto, segundo, milisegundo); trabaja de la misma forma que el método Date.UTC: año y mes son obligatorios y el resto opcionales. El año ha de ser mayor que 1900. Los meses van de 0 a 11 y los días de 1 a 31. Devuelve el número de milisegundos transcurridos desde el 1 de enero de 1970 a las 00:00:00 UTC hasta el instante UTC indicado en los argumentos. Ejemplo:

```
var miFecha = new Date(1970, 0, 2, 0, 0, 0);

alert('Milisegundos desde 1 de enero de 1970 son: '+(miFecha-0));
```

Escribimos una operación aritmética para forzar a que el método toString devuelva el valor numérico en lugar de una representación del tipo << Fri Jan 02 1970 00:00:00 GMT+0000 (Hora de verano GMT)>>

MÉTODOS GETTERS DE OBJETOS TIPO DATE

Los objetos de tipo Date disponen de estos métodos:

MÉTODO	UTILIDAD	EJEMPLOS aprenderaprogramar.com

getFullYear()	Devuelve un entero de cuatro dígitos, año correspondiente a la fecha según el tiempo local del sistema.	Suponiendo que tenemos como hora local del sistema UTC-6 si ejecutamos:
		var miFecha = new Date('2050-12-31T23:00:00-0600');
		var miDia = miFecha.getFullYear(); //Tenemos 2050

getUTCFullYear()	Devuelve un entero de cuatro dígitos, año correspondiente a la fecha según el tiempo UTC.	Independientemente de en qué zona horaria estemos si ejecutamos:
		var miFecha = new Date('2050-12-31T23:00:00-0600');
		var miDia = miFecha.getUTCFullYear(); //Tenemos 2051

getMonth() y **getUTCMonth()**	Devuelve un entero (entre 0 y 11) correspondiente al mes según el tiempo local del sistema ó según el tiempo UTC.	Suponiendo que tenemos como hora local del sistema UTC-6 si ejecutamos:
		var miFecha = new Date('2050-12-31T23:00:00-0600');
		var miDia1 = miFecha.getMonth(); //Tenemos 11
		var miDia2 = miFecha.getUTCMonth(); //Tenemos 0

getDate() y **getUTCDate()**	Devuelve un entero, día del mes correspondiente a la fecha según el tiempo local del sistema ó según el tiempo UTC.	Suponiendo que tenemos como hora local del sistema UTC-6 si ejecutamos:
		var miFecha = new Date('2050-12-24T23:00:00-0600');
		var miDia1 = miFecha.getDate(); //Tenemos 24
		var miDia2 = miFecha.getUTCDate(); //Tenemos 25

getDay() y **getUTCDay()**	Devuelve un entero, día de la semana (entre 0 y 6, siendo el 0 el domingo), correspondiente a la fecha según el tiempo local del sistema ó según el tiempo UTC.	Suponiendo que tenemos como hora local del sistema UTC-6 si ejecutamos:
		var miFecha = new Date('2050-12-24T23:00:00-0600');
		var miDia1 = miFecha.getDay(); //Tenemos 6 (sábado)
		var m = miFecha.getUTCDay(); //Tenemos 0 (domingo)

getHours() y **getUTCHours()**	Devuelve un entero (entre 0 y 23), correspondiente a la hora según el tiempo local del sistema ó según el tiempo UTC.	Suponiendo que tenemos como hora local del sistema UTC-6 si ejecutamos:
		var miFecha = new Date('2050-12-24T23:00:00-0600');
		var miDia1 = miFecha.getHours(); //Tenemos 23
		var m = miFecha.getUTCHours(); //Tenemos 5

getMinutes() y **getUTCMinutes()**	Devuelve un entero (entre 0 y 59) correspondiente al minuto según el tiempo local del sistema o según el tiempo UTC.	Suponiendo que tenemos como hora local del sistema UTC-6 si ejecutamos:
		var miFecha = new Date('2050-12-24T23:35:00-0600');
		var miDia1 = miFecha.getMinutes(); //Tenemos 35
		var m = miFecha.getUTCMinutes(); //Tenemos 35

getSeconds() y **getUTCSeconds()**	Devuelve un entero (entre 0 y 59) correspondiente al segundo según el tiempo local del sistema o según el tiempo UTC.	Suponiendo que tenemos como hora local del sistema UTC-6 si ejecutamos:
		var miFecha = new Date('2050-12-24T23:35:15-0600');
		var miDia1 = miFecha.getSeconds(); //Tenemos 15
		var m = miFecha.getUTCSeconds(); //Tenemos 15

MÉTODO	UTILIDAD	EJEMPLOS aprenderaprogramar.com
getMilliseconds() y getUTCMilliseconds()	Devuelve un entero (entre 0 y 999) correspondiente a los milisegundos según el tiempo local del sistema ó según el tiempo UTC.	Sólo útil para aplicaciones muy específicas.

getTime()	Devuelve un entero, número de milisegundos transcurridos desde el 1 de enero de 1970 00:00:00 UTC. El valor es negativo para fechas anteriores a esta fecha.	var miFecha = new Date('1970-01-02T00:00:00+0000');
		var milisec = miFecha.getTime();
		alert('Tenemos: '+milisec); // Tenemos 86400000

getTimezoneOffset()	Devuelve un entero, valor en minutos de la operación tiempo UTC – tiempo local del sistema. Valor negativo para zonas horarias + y positivo para zonas horarias –.	Si tenemos hora local del sistema GMT+0100:
		var desfase = new Date().getTimezoneOffset();
		alert('Tenemos: '+desfase); // Tenemos -60

MÉTODOS SETTERS DE OBJETOS TIPO DATE

Los objetos de tipo Date disponen de estos métodos para establecer los datos correspondientes a una fecha:

MÉTODO	UTILIDAD	EJEMPLOS aprenderaprogramar.com
setFullYear()	Establece un entero de cuatro dígitos, año correspondiente a la fecha según el tiempo local del sistema.	miFecha.setFullYear(2050);
setUTCFullYear()	Establece un entero de cuatro dígitos, año correspondiente a la fecha según el tiempo UTC.	miFecha.setUTCFullYear(2050);

setMonth() y setUTCMonth()	Establece un entero (entre 0 y 11) correspondiente al mes según el tiempo local del sistema ó según el tiempo UTC.	miFecha.setMonth(11); //El mes es diciembre
		miFecha.setUTCMonth(11); //El mes es diciembre

setDate() y setUTCDate()	Establece un entero, día del mes correspondiente a la fecha según el tiempo local del sistema ó según el tiempo UTC.	miFecha.setDate(24); miFecha.setUTCDate(24);
setHours() y setUTCHours()	Establece un entero (entre 0 y 23), correspondiente a la hora según el tiempo local del sistema ó según el tiempo UTC.	miFecha.setHours(23); miFecha.setUTCHours(23);

MÉTODO	UTILIDAD	EJEMPLOS aprenderaprogramar.com
setMinutes() y setUTCMinutes()	Establece un entero (entre 0 y 59) correspondiente al minuto según el tiempo local del sistema o según el tiempo UTC.	miFecha.setMinutes(35); miFecha.setUTCMinutes(35);
setSeconds() y setUTCSeconds()	Establece un entero (entre 0 y 59) correspondiente al segundo según el tiempo local del sistema o según el tiempo UTC.	miFecha.setSeconds(15); miFecha.setUTCSeconds(15);
setMilliseconds() y setUTCMilliseconds()	Establece un entero (entre 0 y 999) correspondiente a los milisegundos según el tiempo local del sistema ó según el tiempo UTC.	Sólo útil para aplicaciones muy específicas.
setTime()	Establece un entero, número de milisegundos transcurridos desde el 1 de enero de 1970 00:00:00 UTC. El valor es negativo para fechas anteriores a esta fecha.	miFecha.setTime(86400000); //La fecha es 2 de enero de 1970 00:00:00 +0000

PARADOJAS CON FECHAS Y RECOMENDACIÓN

Suponiendo que estamos en México D.F. (zona horaria) UTC-6 si ejecutamos:

var miFecha1 = new Date('2050-12-24T23:00:00-0600');

var miDia = miFecha.getDate(); da lugar a que miDia contenga 24

Suponiendo que estamos en Lisboa (zona horaria UTC+0000) si ejecutamos:

var miFecha1 = new Date('2050-12-24T23:00:00-0600');

var miDia = miFecha.getDate();da lugar a que miDia contenga 25

¿Por qué en un caso obtenemos 24 y en otro 25? Primeramente partimos de la fecha '2050-12-24T23:00:00-0600', esta fecha son las 5 de la mañana del 25 de diciembre de 2050 hora UTC. El día del mes expresado como día local depende de en qué zona del planeta estemos: si estamos en México D.F. a esa hora UTC serán las 23:00 del 24 de diciembre, por ello el método getDate() devolvería 24 en un computador situado en México D.F. Si estamos en Lisboa, a esa hora UTC son las 5 de la mañana del 25 de diciembre y por ello el método getDate() devolvería 25 en un computador situado en Lisboa. Cuando decimos situado nos referimos a que tenga como hora local del sistema la hora local del sitio donde está ubicado el computador (aunque no necesariamente tendría por qué ser así).

Esta aparente paradoja (que no es tal) podría traernos complicaciones si no tenemos en cuenta los detalles. A modo de recomendación, recomendamos trabajar siempre bajo la referencia UTC y hacer conversiones a fechas locales de forma controlada.

EJERCICIO

Crea un documento HTML que conste de un título h1 con el texto <<Calendario>>, y un div central de 400 por 400 px con el borde marcado y márgenes de 100px en todas direcciones. Dentro del div central crea una tabla de 7 columnas por 7 filas (total 49 celdas) con ancho de tabla 300 píxeles y tamaño de fuente en la tabla 24 píxeles. La primera columna corresponderá a lunes y la última a domingo. Usa un método JavaScript para determinar el mes actual. Mediante código JavaScript, haz que aparezca dinámicamente como texto encima de la tabla el mes y año de que se trate (por ejemplo <<MAYO DE 2050>>). Haz que cada celda de la primera fila se rellene indicando el día de la semana (Lu – Ma – Mi – Ju – Vi –Sa –Do). Haz que la tabla se rellene dinámicamente (al cargar la página) con:

a) Espacio en blanco si no corresponde ningún día.

b) El número del día del mes según corresponda (28, 30 ó 31 días según de qué mes se trate).

El aspecto, suponiendo que te encuentras en el mes de mayo de 2050, sería el siguiente:

MAYO DE 2050

Lu	Ma	Mi	Ju	Vi	Sa	Do
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Ten en cuenta que debe generarse el calendario del mes en que te encuentres según la hora local del sistema (de tu computador).

Para comprobar si tus respuestas y código son correctos puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU01163E

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=78&Itemid=206