Objetos nativos. Date

- C.F.G.S. DAW
- 6 horas semanales
- Mes aprox. de impartición: Nov
- iPasen - cjaedia071@g.educaand.es

Carmelo José Jaén Díaz

Índice

OBJETIVO

- Aprender cómo se puede manejar el tiempo en JavaScript.
- Saber cómo se programa la ejecución de funciones de forma puntual y periódica en el tiempo.
- Registrar en el front-end información de navegación, usuario, etc., mediante cookies o mediante el almacenamiento local que ofrece el nuevo estándar HTML5.
- Profundizar en el concepto de objeto.
- Conocer y manejar funciones relativas al lenguaje sobre arrays, strings, números.

GLOSARIO

Backtracking. Estrategia utilizada en algoritmos que resuelven problemas que tienen ciertas restricciones. Este término fue creado por primera vez por el matemático D. H. Lehmer en la década de los cincuenta.

BOM (Browser Object Model). Convención específica implementada por los navegadores para que JavaScript pudiese hacer uso de sus métodos y propiedades de forma uniforme.

Expresión regular. Secuencia de caracteres que forman un patrón determinado, generalmente un patrón de búsqueda.

NaN. Propiedad que indica Not a Number (valor no numérico).

Objeto window. Aquel que soportan todos los navegadores y que representa la ventana del navegador. Se estudiará en profundidad en capítulos posteriores.

URI (Uniform Resource Identifier). Cadena de caracteres que identifica un recurso en una red de forma unívoca. Una URI puede ser una URL, una URN o ambas.

GLOSARIO

URN. Localizador de recursos en la web que funciona de forma parecida a una URL, pero su principal diferencia es que no indica exactamente dónde se encuentra dicho objeto.

INTRODUCCIÓN

El **objeto nativo *Date*** se utiliza para trabajar con fechas y es uno de los más odiados entre los estudiantes... lo sé. Entender que está formado por milisegundos desde **el día 1 de enero de 1970** y tener que formatearlo siempre que queremos mostrarlo es un poco rollo. ¡Pero es lo que hay!

En primer lugar, hay que entender todos los modos de crear un objeto *Date*. Aquí tenéis varios modos:

```
new Date()
new Date(milisegundos)
new Date(cadenaFecha)
new Date(año_num,mes_num,dia_num
[,hor_num,min_num,seg_num,mils_num])
// ¡Recuerda que todo el código entre corchetes es opcional!
```

INTRODUCCIÓN

Claro está, dependiendo de lo que introducimos entre paréntesis podemos obtener objetos con la fecha actual, con fechas elegidas por nosotros, con fechas formateadas a nuestro gusto...

A continuación, veremos las siguientes cuestiones:

- Cómo crear un objeto date con la fecha actual.
- Crear fechas con cadenas como parámetro.
- Crear fechas con milisegundos como parámetro.
- Crear fechas con números como parámetros.
- Visualizar fechas con diferentes formatos: toString, toUTCString, toDateString.

INSTANCIACIÓN DE *Date*.
Creación de fecha actual

IMPORTANTE:

Los meses comienzan en 0. Los días de la semana también, siendo el 0 el domingo, el 1 el lunes...

No escribir los días o meses menores que 10 sin el 0, es decir, el formato ha de ser XX.

FORMA DE INSTANCIACIÓN

```
let actual = new Date();
alert(actual);
```

INSTANCIACIÓN DE *Date*.
Creación de fecha con cadenas

NOTA:

La forma más completa se obtiene al pasar como argumento todos los datos (en inglés) de la fecha.

FORMA DE INSTANCIACIÓN

```
let d1 = new Date("Mon Nov 25 2024 09:23:45 GMT +0100 (W. Europe
Standard Time)");
//Ignora si el día de la semana está mal o el contenido entre
paréntesis

alert (d1);
```

INSTANCIACIÓN DE *Date*.
Creación de fecha con cadenas

FORMA DE INSTANCIACIÓN

```
let d2 = new Date("October 25, 2024 10:30:00");

alert (d2);
```

INSTANCIACIÓN DE *Date*.
Creación de fecha con cadenas

NOTA:

En estos formatos es posible usar comas, ya que se ignoran.

FORMA DE INSTANCIACIÓN

```
let d3 = new Date("January 25 2024");
let d4 = new Date("Jan 25 2024"); //También "25 Jan 2024"

alert (d3);
alert (d4);
```

INSTANCIACIÓN DE *Date*.
Creación de fecha con cadenas

FORMA DE INSTANCIACIÓN

```
let d5 = new Date("2024-05-12T12:34:25"); // YYYY-MM-DD
let d6 = new Date("2024-05-12"); // YYYY-MM-DD
let d7 = new Date("2024/05/12"); // YYYY/MM/DD o DD/MM/YYYY

alert (d5);
alert (d6);
alert (d7);
```

INSTANCIACIÓN DE *Date*.
Creación de fecha con cadenas

FORMA DE INSTANCIACIÓN

```
let d8 = new Date("2024-05"); //El día se sustituye por 1
let d9 = new Date("2024"); //El día y el mes se sustituyen por 1

alert (d8);
alert (d9);
```

INSTANCIACIÓN DE *Date*.
Creación de fecha con milisegundos

NOTA:

Se le pasa como argumento el número de milisegundos que han transcurrido desde el 01/01/1970.

FORMA DE INSTANCIACIÓN

```
let dms = new Date(86400000);
alert ("Fecha en ms: "+dms);
```

INSTANCIACIÓN DE *Date*.
Creación de fecha con números

RECUERDA:

Los meses comienzan en 0. Los días de la semana también, siendo el 0 el domingo, el 1 el lunes...

No escribir los días o meses menores que 10 sin el 0, es decir, el formato ha de ser XX.

FORMA DE INSTANCIACIÓN

```
let fechaLargo = new Date(2024,11,10,14,30,25); // Fecha y hora
let fechaCorto = new Date(2024,11,10); // hora establecida a 00:00:00

alert ("Fecha largo: "+fechaLargo+ " Fecha corto: "+fechaCorto);
```

VISUALIZACIÓN DE FECHAS

```
var fecha = new Date();

alert ("Fecha: "+fecha);
alert ("toString: "+fecha.toString());
alert ("toUTCString: "+fecha.toUTCString());
alert ("toDateString: "+fecha.toDateString());

//Probar para ver los distintos formatos de salida
```

MÉTODOS DE *Date*

A continuación, vamos a ver cómo extraer toda la información que almacena un objeto *Date*. Es decir, cómo obtener los días, horas, minutos, segundos... de este tipo de datos. Y cómo modificarlos uno a uno.

MÉTODOS DE TIPO *GET*

- *getDay*
- *getDate*
- *getMonth*
- *getFullYear*
- *getHours*
- *getMinutes*
- *getSeconds*
- *getMilliseconds*
- *getTime*

¡Recuerda! Cuando utilizamos *getDay* sobre un objeto *Date* no obtenemos el día del mes, sino el día de la semana, siendo el 0 el domingo.

Podemos encontrar todos los métodos de *Date* en W3Schools.com>JavaScript>JS References.

MÉTODOS DE TIPO *SET*

- *setDay*
- *setDate*
- *setMonth*
- *setFullYear*
- *setHours*
- *setMinutes*
- *setSeconds*
- *setMilliseconds*
- *setTime*

MÉTODOS DE *Date*.
Obtención de fecha: *getDay*

Método: getDay()

Finalidad: devuelve el día de la semana, siendo el 0 el domingo, 1 el lunes, etc.
Al devolver un número, si queremos obtener su correspondiente nombre del día podríamos crear un array de nombres de días y vincular posiciones.

Ejemplo de uso:

```
let fc = new Date("October 1, 2024 10:30:20");

alert(fc.getDay());
```

MÉTODOS DE *Date*.
Obtención de fecha: *getDate*

Método: getDate()

Finalidad: devuelve el día del mes.

Ejemplo de uso:

```
let fc = new Date("October 1, 2024 10:30:20");

alert(fc.getDate());
```

MÉTODOS DE *Date*.
Obtención de fecha: *getMonth*

Método: getMonth()

Finalidad: devuelve el mes, siendo el 0 enero, 1 febrero, etc.

Al devolver un número, si queremos obtener su correspondiente nombre del mes podríamos crear un array de nombres de meses y vincular posiciones.

Ejemplo de uso:

```
let fc = new Date("October 1, 2024 10:30:20");

alert(fc.getMonth());
```

MÉTODOS DE *Date*.
Obtención de fecha: *getFullYear*

Método: getFullYear().

Existe getYear() pero está obsoleta.

Finalidad: devuelve el año.

Ejemplo de uso:

```
let fc = new Date("October 1, 2024 10:30:20");

alert(fc.getFullYear());
```

MÉTODOS DE *Date*.
Obtención de hora: *getHours*

Método: getHours()

Finalidad: devuelve la hora.

Ejemplo de uso:

```
let fc = new Date("October 1, 2024 10:30:20");

alert(fc.getHours());
```

MÉTODOS DE *Date*.
Obtención de hora: *getMinutes*

Método: getMinutes()

Finalidad: devuelve los minutos.

Ejemplo de uso:

```
let fc = new Date("October 1, 2024 10:30:20");

alert(fc.getMinutes());
```

MÉTODOS DE *Date*.
Obtención de hora: *getSeconds*

Método: getSeconds()

Finalidad: devuelve los segundos.

Ejemplo de uso:

```
let fc = new Date("October 1, 2024 10:30:20");

alert(fc.getSeconds());
```

MÉTODOS DE *Date*.
Obtención de hora: *getMilliseconds*

Método: getMilliseconds()

Finalidad: devuelve los milisegundos.

Ejemplo de uso:

```
let fc = new Date("October 1, 2024 10:30:20");

alert(fc.getMilliseconds());
```

MÉTODOS DE *Date*.
Obtención de milisegundos: *getTime*

Método: getTime()

Finalidad: devuelve los milisegundos desde el 1 de enero de 1970.
Una forma de obtener el tiempo que ha transcurrido entre dos fecha es obtener los milisegundos que han pasado desde el 1 de enero de 1970 en ambas, restarlos y dividir por el correspondiente número de años, meses, días, horas, minutos y segundos.

Ejemplo de uso:

```
alert("Ms desde 1/1/1970: "+fc.getTime());
```

MÉTODOS DE *Date*.
Modificación de fecha: *setDate*

Método: setDate(<número>)

Finalidad: modifica el día del mes.

Ejemplo de uso:

```
let fc = new Date("October 1, 2024 10:30:20");

fc.setDate(31);

//¿Recuerdas cómo obtener dicho dato y mostrarlo en una ventana emergente?
```

MÉTODOS DE *Date*.
Modificación de fecha: *setMonth*

Método: setMonth(<número>)

Finalidad: modifica el mes.

Ejemplo de uso:

```
let fc = new Date("October 1, 2024 10:30:20");

fc.setMonth(11); //Hace referencia a diciembre

//¿Recuerdas cómo obtener dicho dato y mostrarlo en una ventana emergente?
```

MÉTODOS DE *Date*.
Modificación de fecha: *setFullYear*

Método: setFullYear(<número>)

Finalidad: modifica el año.

Ejemplo de uso:

```
let fc = new Date("October 1, 2024 10:30:20");

fc.setFullYear(2031);

//¿Recuerdas cómo obtener dicho dato y mostrarlo en una ventana emergente?
```

MÉTODOS DE *Date*.
Modificación de hora: *setHours*

Método: setHours(<número>)

Finalidad: modifica la hora.

Ejemplo de uso:

```
let fc = new Date("October 1, 2024 10:30:20");

fc.setHours(23);

//¿Recuerdas cómo obtener dicho dato y mostrarlo en una ventana emergente?
```

MÉTODOS DE *Date*.
Modificación de hora: *setMinutes*

Método: setMinutes(<número>)

Finalidad: modifica los minutos.

Ejemplo de uso:

```
let fc = new Date("October 1, 2024 10:30:20");

fc.setMinutes(59);

//¿Recuerdas cómo obtener dicho dato y mostrarlo en una ventana emergente?
```

MÉTODOS DE *Date*.
Modificación de hora: *setSeconds*

Método: setSeconds(<número>)

Finalidad: modifica los segundos.

Ejemplo de uso:

```
let fc = new Date("October 1, 2024 10:30:20");

fc.setSeconds(59);

//¿Recuerdas cómo obtener dicho dato y mostrarlo en una ventana emergente?
```

MÉTODOS DE *Date*.
Modificación de hora a partir de milisegundos

Método: setMilliseconds(<ms>)

Finalidad: modifica la fecha y la hora contando los milisegundos desde el 1/1/1970.

Ejemplo de uso:

```
let fc = new Date("October 1, 2024 10:30:20");

fc.setTime(1987654321987654321);

alert(fc.toString());
```

CÁLCULOS CON FECHAS

EJEMPLO: Obtener la fecha correspondiente al transcurso de tres meses a partir de una fecha dada.

Implementación:

```
let fc = new Date("October 1, 2024 10:30:20");

fc.setMonth(fc.getMonth()+3);

alert(fc.toString());
```