TEMA 8

OTROS OBJETOS

OBJETO DATE

Manipular fechas es una tarea complicada, no solo porque las fechas están compuestas por varios valores que representan diferentes componentes, sino porque estos componentes están relacionados. Si un valor sobrepasa su límite, afecta al resto de los valores de la fecha. El límite para minutos y segundos es 60, pero el límite en las horas es 24, y cada mes tiene un número diferente de días. Además, existen diferentes zonas horarias, cambios de horarios según la estación, etc. Para simplificar el trabajo de los desarrolladores, JavaScript define un objeto llamado Date.

El objeto Date almacena una fecha y se encarga de mantener los valores dentro de sus límites. La fecha en estos objetos se almacena en milisegundos, lo que nos permite realizar operaciones entre fechas, calcular intervalos, etc. Concretamente, representa los milisegundos transcurridos desde el 1 de Enero de 1970.

1. CONSTRUCTOR DE DATE

Para crear un nuevo objeto Date se lo instancia con new Date(valor), es decir Date es el constructor con el que crearemos un objeto de este tipo. Si no le pasamos ningún dato, nos devolverá la fecha actual del sistema.

```
<body>
    <h1>Son las  <script> document.write(new Date())</script></h1>
</body>
```

El atributo valor se puede declarar como una cadena de caracteres o como los componentes de una fecha separados por comas, en este orden: año, mes, día, horas, minutos, segundos y milisegundos. Si le pasamos un valor numérico, representará los milisegundos transcurridos desde el 1 de enero de 1970. Veamos como ejemplo que si lo que pasamos es el 0 nos devolverá precisamente esa fecha y si pasamos cualquier otro valor nos calculará la fecha que representa.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Document</title>
8  </head>
9  <body>
10     <h1>Son las  <script> document.write(new Date(0))</script></h1>
11 </body>
12 </html>
```

Los valores enteros que representan la cantidad de milisegundos transcurridos desde el inicio de 1970, también denominados timestamp, puede ser negativo si queremos representar una fecha anterior.

Si usamos un conjunto de argumentos separados por comas, lo que haremos será escribirlo de la siguiente manera:

new Date (año, mes, fecha, horas, minutos, segundos, ms)

Crea una fecha con los componentes pasados como argumentos en la zona horaria local. Solo los dos primeros son obligatorios. Y los requisitos son los siguientes:

- El año debe tener 4 dígitos, si ponemos dos serán considerado “19xx”, pero 4 dígitos es lo firmemente sugerido.
- La cuenta del mes empieza en el 0 (enero), y termina en 11 (diciembre).
- El parámetro día sería el día del mes, si no ponemos nada, se asume que es 1.
- Si los siguientes parámetros están ausentes, se asumen sus valores iguales a 0.

Sería por tanto equivalente escribir lo siguiente:

new Date (2011, 0, 1, 0, 0, 0, 0);
new Date (2011, 0, 1);

También podemos utilizar Date como si fuera una función, regresando una cadena que representa la fecha y hora actual, exactamente como lo hace new Date(0). Veámoslo en el siguiente ejemplo:

```
<body>
    <script>
        document.write(Date().toString())
        document.write(Date())
    </script>
```

Debemos fijarnos que en este caso no generamos objeto, es decir, no utilizamos la palabra new, manejamos Date como si fuera una función.

2. METODOS ASOCIADOS A DATE

Los objetos Date ofrecen los siguientes métodos para obtener los componentes de la fecha, como el año o el mes.

- **now():** Devuelve el valor numérico correspondiente al actual número de milisegundos transcurridos desde el 1 de enero de 1970, ignorando los segundos intercalares.
- **parse():** transforma la cadena que representa una fecha y devuelve el número de milisegundos transcurridos desde el 1 de enero de 1970.
- **getFullYear():** este método devuelve un número entero que representa el año (un valor de 4 dígitos).
- **getMonth():** este método devuelve un número entero que representa el mes (un valor de 0 a 11).
- **getDate():** Este método devuelve un número entero que representa el día del mes (un valor de 1 a 31).
- **getDay():** este método devuelve un número entero que representa el día de la semana (un valor de 0-domingo a 6-sábado).
- **getHours():** Este método devuelve un número que devuelve un número que representa las horas (un valor de 0 a 23).
- **getMinutes():** Este método devuelve un número entero que representa los minutos (un valor de 0 a 59).
- **getSeconds():** este método devuelve un número entero que representa los segundos (un valor de 0 a 59).
- **getMilliseconds():** este método devuelve un número entero que representa los milisegundos (un valor de 0 a 999).

- **getTime():** este método devuelve un número entero en milisegundos que representa el intervalo desde el 1 de enero de 1970 hasta la fecha.

Los objetos Date también incluyen métodos para modificar los componentes de la fecha.

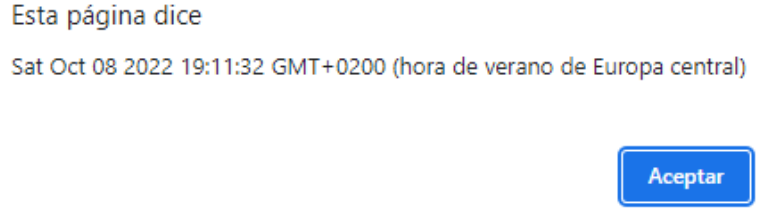
- **setFullYear(año):** este método especifica el año (un valor de 4 dígitos).
- **setMonth(mes):** este método especifica el mes (un valor de 0 a 11).
- **setDate(dia):** este método especifica el día (un valor de 1 a 31).
- **setHours(horas):** este método especifica la hora (un valor de 0 a 23).
- **setMinutes(minutos):** este método especifica los minutos (un valor de 0 a 59).
- **setSeconds(segundos):** este método especifica los segundos (un valor de 0 a 59).
- **setMilliseconds(milisegundos):** este método especifica los milisegundos (un valor de 0 a 999).

Veamos un ejemplo donde mostraremos la fecha con el formato que utilizamos normalmente al mostrar la información asociada a una fecha.

```
<script>
let fecha = new Date()
let fechaFormato = fecha.getDate()+"/"+(fecha.getMonth()+1)+"/"+fecha.getFullYear()
alert(fechaFormato)
</script>
```

Veamos otro ejemplo de cómo modificar la fecha, escribiremos el resultado para ver que cuando se modifica el día, también lo hace el día de la semana que será.

```
<script>
    let fecha = new Date()
    alert(fecha)
    fecha.setDate(8)
    alert(fecha)
</script>
```



Como podemos ver el día de la semana del 8 de octubre que será sábado se modifica al modificar el día de la semana asociada a una fecha.

OBJETO WINDOW

Cada vez que abrimos el navegador o iniciamos una nueva pestaña, se crea un objeto global llamado window para referenciar la ventana del navegador y proveer algunas propiedades y métodos esenciales. El objeto se almacena en una propiedad del objeto global de JavaScript llamada window. A través de esta propiedad podemos conectarnos con el navegador y el documento desde nuestro código.

El objeto window a su vez incluye otros objetos con los que se provee información adicional relacionada con la ventana y el documento. Veamos algunos:

- **Location:** Esta propiedad contiene un objeto **Location** con información acerca del origen del documento. También se puede usar como una propiedad para declarar o devolver la URL del documento (por ejemplo, **window.location** =http://www.fpmarco.com).
- **Navigator**: Esta propiedad contiene un objeto Navigator con información sobre la aplicación y el dispositivo.
- **Document**: Esta propiedad contiene un objeto Document, que provee a los objetos que representan los elementos HTML en el documento.

Además de estos valiosos objetos, el objeto Window también ofrece sus propias propiedades y métodos. Los siguientes son los que más se usan:

- **innerWidth:** Esta propiedad devuelve el ancho de la ventana en píxeles.
- **innerHeight:** Devuelve la altura de la ventana en píxeles.
- **scrollX**: esta propiedad devuelve el número de píxeles en los que el documento se ha desplazado horizontalmente.
- **scrollY**: esta propiedad devuelve el número de píxeles en los que el documento se ha desplazado verticalmente.
- **alert(valor)**: este método muestra una ventana emergente en la pantalla que muestra el valor entre paréntesis.
- **confirm(mensaje):** este método es similar a alert(), pero ofrece dos botones, Aceptar y Cancelar, para que el usuario elija qué hacer. El método devuelve true o false, según la respuesta del usuario.
- **prompt(mensaje):** este método muestra una ventana emergente con un campo de entrada para permitir al usuario introducir un valor. El método devuelve el valor que inserta el usuario.
- **setTimeout(función, milisegundos):** este método ejecuta la función especificada en el primer atributo cuando haya pasado el tiempo especificado por el segundo atributo.
- **clearTimeout():** sirve para cancelar el proceso anterior, setTimeout.
- **setInterval(función, milisegundos):** es similar a setTimeout(), pero llama a la función constantemente.
- **clearInterval():** sirve para cancelar el método anterior.
- **OPEN(URL, ventana, parámetros):** Este método abre un documento en una nueva ventana. El atributo URL es la dirección del documento que queremos abrir, el atributo ventana es el nombre de la ventana donde queremos mostrar el documento, si no se especifica se abrirá en una nueva ventana; y el atributo parámetros serán de configuración separados por comas que configuran las características de la ventana ("**resizable=no,scrollbars=no**"). También existe **close()** para cerrar una ventana abierta con open.

El objeto window controla aspectos de la ventana, su contenido, y los datso asociados a la misma, como la ubicación del documento actual, el tamaño, desplazamiento, etc. Veamos algún ejemplo:

```
<body>
    <script src="localizacion.js">
        </script>
    <input type="button" onclick="abrirPagina()" value="Pulsa">
</body>
```

Generamos un archivo html donde asociamos a un botón un evento de click que ejecutará la función abrirPagina cuando lo pulsemos, veamos cómo esta función abrirá la página de nuestro centro.

```
function abrirPagina(){
    window.location="https://fpmarco.com/formacion-profesional/"
}
```

Además de asignar una nueva URL a la propiedad location, también podemos manipular la ubicación desde los métodos provistos por el objeto Location.

- **Assign(URL):** Este método le pide al navegador que cargue el documento en la ubicación especificada por el atributo URL.
- **Replace(URL):** Este método le pide al navegador que reemplace el documento actual con el documento en la ubicación indicada por el atributo URL. Difiere del método assign() en que no agrega la URL al historial del navegador.
- **Reload(valor):** Este método le pide al navegador que actualice el documento actual. Acepta un valor booleano que determina si el recurso se tiene que descargar desde el servidor o se puede cargar desde el caché del navegador (true o false).

El siguiente ejemplo actualiza la página cuando el usuario pulsa un botón. Como podemos ver en el ejemplo no se escribe la propiedad window. El objeto window es un objeto global y, por tanto, el intérprete infiere que las propiedades y los métodos pertenecen a ese objeto. Por eso nunca escribimos window.alert() cuando queremos que nos aparezca la ventana emergente.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script>
        function recargar(){
            location.reload()
        }
    </script>
</head>
<body>
    <script>alert("Estoy entrando")</script>
    <input type="button" onclick="recargar()" value="Recargar">
</body>
</html>
```

Veamos un ejemplo de cómo funciona setTimeout, recordar que lo que hace es ejecutar la función que especifiquemos cuando pase el tiempo adecuado, el segundo argumento es con milisegundos.

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script>
        function realizar(){
            let hoy =new Date()
            let tiempo = hoy.toTimeString()
            alert(tiempo)
        }
    </script>
</head>
<body>
    <input type="button" onclick="setTimeout(realizar,5000)" value="Pulsa aquí">
</body>
</html>
```

OBJETO DOM

Como hemos dicho, JavaScript es orientado a objetos, como podemos ver todo en JavaScript se define como un objeto, y esto incluye los elementos de un documento. Cuando se carga un documento HTML, el navegador crea una estructura interna para procesarlo. La estructura se llama DOM (Document Object Model) y está compuesta por múltiples objetos de tipo Element (u otros más específicos que heredan de Element), que representan cada elemento en el documento.

Los objetos Element mantienen una conexión permanente con los elementos que representan. Cuando se modifica un objeto, su elemento también se modifica y el resultado se muestra en pantalla. Para ofrecer acceso a estos objetos y permitirnos alterar sus propiedades desde nuestro código JavaScript, los objetos se almacenan en un objeto llamado Document que se asigna a la propiedad document del objeto Window.

Entre otras alternativas, el objeto Document incluye las siguientes propiedades para ofrecer acceso rápido a los objetos Element que representan los elementos más comunes del documento.

- **forms:** Esta propiedad devuelve un array con referencias a todos los objetos Element que representan los elementos <form> en el documento.
- **images:** Esta propiedad devuelve un array con referencias a todos los objetos Element que representa los elementos <img> en el docuemnto.
- **links:** Esta propiedad devuelve un array con referencias a todos los objetos Element que representan los elementos <a> en el documento.