C.F.G.S.: DESARROLLO DE APLICACIONES WEB
Módulo: DESARROLLO WEB EN ENTORNO CLIENTE

05 OBJETOS STRING, DATE, MATH, NUMBER

Otros objetos predefinidos del núcleo de JavaScript

El objeto Date

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Date

https://www.w3schools.com/js/js_dates.asp

- Sirve para manejar fechas
- En JavaScript las fechas se expresan como el número de milisegundos transcurridos desde el 1 de enero de 1970 a las 00:00:00 UTC (Tiempo Universal Coordinado) Esta fecha se conoce como origen o época UNIX .
- Para crear un objeto de este tipo :

Creating Date Objects

Date objects are created with the `new Date()` constructor.

There are **9 ways** to create a new date object:

```
new Date()
new Date(date string)

new Date(year,month)
new Date(year,month,day)
new Date(year,month,day,hours)
new Date(year,month,day,hours,minutes)
new Date(year,month,day,hours,minutes,seconds)
new Date(year,month,day,hours,minutes,seconds,ms)

new Date(milliseconds)
```

let fecha = new Date();

fecha contendrá la fecha y hora del sistema

let fecha = new Date(*milisegundosDesdeEpocaUNIX*);

construye la fecha correspondiente

let fecha = new Date("*cadena año-mes-día*");

construye la fecha correspondiente

var fecha = new Date(año,mes,dia[,hora,minuto,segundo,milisegundo]);

Donde todos los parámetros son números enteros.

año debe expresarse con 4 cifras.

mes es un entero entre 0 (enero) y 11 (diciembre).

día es un entero entre 1 y 31.

hora, minuto, segundo, milisegundo no son obligatorios.

```
let fechaHoy = new Date();
let fecha1 = new Date (1398984992375);
let fecha2 = new Date(1999,9,7);
let fecha3 = new Date("2023-10-18")
alert (fechaHoy);
alert (fecha1);
alert (fecha2);
alert (fecha3);
```

EJ: Comprobar que sucede en estos casos

Fecha=new Date (2022,12,45)
Fecha=new Date (2022,9)
Fecha=new Date (2022,12, 'a')

Para saber si una fecha se ha construido correctamente podemos preguntar por isNaN(Fecha) de devolverá false o true dependiendo de que lo haya sido o no.

El objeto *Date* no presenta ninguna propiedad pero tiene una serie de métodos que pueden dividirse en tres subconjuntos:

- Métodos de lectura: empiezan con el prefijo *get*. Consultan las diferentes partes de una instancia del objeto Date
- Métodos de escritura: empiezan con el prefijo *set*. Inicializan o modifican las diferentes partes de una instancia del objeto Date

- Métodos de conversión: convierten objetos de tipo Date en cadenas de texto o milisegundos

- **getDate()** Devuelve el día del mes entre 1 y 31.

- **getDay()** Devuelve el día de la semana (0-6). El 0 corresponde al domingo y el 6 al sábado. Atención, el day no se puede establecer, sólo leer; en otras palabras, no existe setDay()

- **getMonth()** Devuelve el mes (0-11).

- **getFullYear()** Devuelve el año expresado con 4 dígitos.

- **getHours()** Devuelve la hora (0-23).

- **getMinutes()** Devuelve los minutos (0-59).

- **getSeconds()** Devuelve los segundos (0-59).

- **getMiliseconds()** Devuelve los milisegundos (0-999).

```
var fechaHoy = new Date();
alert ('Hoy es '+fechaHoy.getDate()+', son las '+fechaHoy.getHours()+
      ' horas y '+fechaHoy.getMinutes()+' minutos');
```

- **getTime()** Devuelve los milisegundos transcurridos desde la época UNIX hasta la fecha que se aplica

```
alert(fechaHoy.getTime());
```

- **setDate()** Establece el día del mes.

- **setMonth()** Establece el mes (0-11).

- **setFullYear()** Establece el año expresado con 4 dígitos.

- **setHours()** Establece la hora (0-23).
- **setMinutes()** Establece los minutos (0-59).

- **setSeconds()** Establece los segundos (0-59).

- **setMiliseconds()** Establece los milisegundos (0-59).

- **setTime()** Establece la fecha como el número de milisegundos transcurrido desde la época UNIX

```
var fecha=new Date();
fecha.setDate(19);
fecha.setMonth(5);
fecha.setFullYear(2023);
alert (fecha);
```

- **toDateString()** Convierte la fecha del objeto Date en una cadena de caracteres
- **toTimeString()** Convierte el tiempo del objeto Date en una cadena de caracteres
- **toLocaleDateString()**Devuelve una cadena de la fecha en el formato de idioma local que tenga configurado el cliente.
- **toLocaleTimeString()**Devuelve una cadena de la hora (hora, minuto, segundo) en el formato de idioma local que tenga configurado el cliente.

Ejemplo:

```
Fecha=new Date (2022,9,8,11,20,22);
alert (Fecha.toDateString()) ; // Tue Oct 08 2022
alert (Fecha.toTimeString()) ; // 11:20:22 GMT+0200
alert (Fecha.toLocaleDateString());//     8/10/2022
alert (Fecha.toLocaleTimeString());//     11:20:22
```

VER EJEMPLOS FECHAS ejemplo 01, ejemplo 02, ejemplo 03

El objeto String

- En JavaScript cualquier cadena de caracteres tiene asociado un objeto de tipo String.
- Su única propiedad es *length* que contiene el número de caracteres que componen la cadena

```
var texto =  'pepe';
alert (texto.length) // 4
```

- Podemos acceder a cada carácter de un string con su posición (índice)

```
alert (texto[2]) // p
```

Los métodos más interesantes del objeto String

- **concat(*cadena1*,…,*cadenaN*)**
  - Concatena *cadena1*, …, *cadenaN* a continuación de la cadena pero la cadena no se queda modificada

```
var otro = 'luis';
alert (texto.concat(otro)); // pepeluis
alert (texto)); // pepe
```

- **indexOf(*aguja*[,*inicio*])**
  - Devuelve la primera posición desde la izquierda en la que se encuentra la cadena *aguja* dentro de la cadena empezando a buscar por la posición *inicio*.
  - Si no se encuentra devuelve -1.

```
alert (texto.indexOf('e')); // 1
alert (texto.indexOf('e',2)); // 3
```

- **lastIndexOf(*aguja*[,*fin*])**
  - Como el anterior pero buscando desde la derecha y comenzando por *fin*

```
alert (texto.lastIndexOf('e')); // 3
```

- **split (separador)**
  - Devuelve un array cuyos elementos son los fragmentos de la cadena comprendidos entre el carácter o cadena *separador*.
  - Puede considerarse el método contrario de otro llamado join que está disponible en los datos de tipo Array

```
var nombres = 'Pepe, Luis, Juan';
alert (nombres.split(',')[1]);   // Luis
```

- **slice(*inicio*[,*fin*]), substring (inicio[,fin])**
  - Devuelve los caracteres comprendidos entre *inicio* (incluido) y *fin* (exluido), o hasta el final de la cadena si se omite *fin*.

```
var texto =  'Hola Pepe, qué tal?';
alert(texto.substring(5,8)); // Pep
alert (texto. slice(5,8)); //   Pep
```

- **substr (inicio[,cantidad])**
  - Devuelve *cantidad* número de caracteres a partir de *inicio* (incluido). Si se omite *cantidad*, devuelve hasta el final de la cadena.

```
var texto =  'Hola Pepe, qué tal?';
alert(texto.substr(5,4));  // Pepe
```

- **toLowerCase()**
  - *Devuelve* la cadena convertida en minúsculas.

```
alert(texto.toLowerCase());// pepe
```

- **toUpperCase()**
  - *Devuelve* la cadena convertida a mayúsculas.

```
alert(texto.toUpperCase());// PEPE
```

- **trim() , trimLeft(), trimRight()**
  - Eliminan los espacios al principio y al final de la cadena, al principio solamente o al final solamente, respectivamente.

- **search(cadena1)**
  - Busca una coincidencia en una cadena y devuelve la posición de la coincidencia
- **replace(cadena1, cadena2)**
  - Busca una coincidencia en una cadena y si existe la reemplaza por otra cadena pasada por parámetro.

```
var texto = 'hola manola';
alert (texto.search('la'));  //2
alert (texto.replace('hola','adios'));
```

El objeto Math

- El objeto Math nos permite realizar operaciones matemáticas: raíces cuadradas, logaritmos, operaciones trigonométricas y, muy importante, obtener números pseudo-aleatorios.
- Math no es un constructor, sino directamente un objeto; en otras palabras, no tiene sentido crear un objeto de tipo Math con la instrucción `new` (produciría un error), sino que debemos utilizar directamente el objeto
- Las propiedades de Math se utilizan para acceder a algunas constantes matemáticas de interés y, obviamente, son de sólo lectura.

http://msdn.microsoft.com/es-es/library/b272f386(v=vs.94).aspx

Propiedad	Descripción
E	Constante de Euler (aprox. 2.718).
LN2	Logaritmo natural de 2.
LN10	Logaritmo natural de 10.
LOG2E	Logaritmo en base 2 de la constante de Euler.
LOG10E	Logaritmo en base 10 de la constante de Euler.
PI	Constante π.
SQRT1_2	Raíz cuadrada de ½.
SQRT2	Raíz cuadrada de 2.

Métodos

- **ceil (valor)** **Devuelve el entero inmediatamente mayor o igual que valor.**
- **floor(valor)** Devuelve el entero inmediatamente menor o igual que *valor*.

- **max(val1, …, valN)** Devuelve el máximo de los valores recibidos como argumentos
- **min(val1, …, valN)** Devuelve el mínimo de los valores recibidos como argumentos
- **pow(base,exponente)** Devuelve el resultado de elevar base a exponente
- **random()** Devuelve un número aleatorio entre 0 y 1, 1 no incluido
- **round(valor)** Devuelve el resultado de redondear valor al entero más próximo
- **sqrt(valor)** Devuelve la raíz cuadrada de valor.

```
alert (Math.PI);   // 3.1415…
alert (Math.ceil (33.3));   // 34
alert (Math.floor(33.3));   // 33
alert(Math.max(2,6,3,7));  // 7
alert(Math.min(2,6,3,7));  // 2
alert(Math.pow(2,4));   // 16
alert(Math.random());
alert(Math.round(33.3));   // 33
alert(Math.sqrt(25));      // 5
```

Devuelven **NaN** cuando no han podido efectuar el cálculo matemático

Otros métodos

Método	**Descripción**
abs(*valor*)	Devuelve el valor absoluto de *valor*, es decir, su valor sin signo.
acos(*valor*)	Devuelve el arcocoseno en radianes (entre 0 y π) de *valor*. Obviamente valor debe estar comprendido entre -1 y 1.
asin(*valor*)	Devuelve el arcoseno en radianes (entre - π/2 y + π/2) de *valor*. Obviamente valor debe estar comprendido entre -1 y 1.
atan(*valor*)	Devuelve el arcotangente en radianes (entre - π/2 y + π/2) de *valor*.
atan2(*ord,abscisa*)	Devuelve el ángulo (entre - π/ y + π) que forma el radiovector de extremo *abscisa*, *ordenada* con el semieje positivo de abscisas. Fíjese en que el orden de los argumentos requiere primero la ordenada y luego la abscisa.
cos(*valor*)	Devuelve el coseno de *valor*, que se considera expresado en radianes.
exp(*valor*)	Devuelve el resultado de elevar la constante de Euler a *valor*.
log(*valor*)	Devuelve el logaritmo natural de *valor*.
sin(*valor*)	Devuelve el seno de *valor*, que se considera expresado en radianes.
tan(*valor*)	Devuelve la tangente de *valor*, que se considera expresado en radianes.

Ejemplo función para generar números aleatorios en un intervalo (extremos incluidos)

```
function aleatorio (min,max){
 return Math.floor(Math.random()*(max - min +1)+min);
}
```

Ejercicio: ¿Valdría esta otra?

```
function aleatorio (min,max){
 return Math.round(Math.random()*(max - min))+min;
}

var a = Array();
var b=Array(0,0,0,0,0,0,0,0,0,0);
for (i=0;i<10000;i++){
a[i]= (aleatorio(1,10));
b[a[i]-1]++;}
alert (b)
```

El objeto Number

Se utiliza principalmente para indicar el máximo y mínimo valor posible que podemos representar con JavaScript e informar al usuario cuando sobrepase esos límites en alguna operación matemática.

Propiedades

MAX_VALUE	Devuelve el mayor número posible en JavaScript
MIN_VALUE	Devuelve el menor número posible en JavaScript
NaN	Representa el valor especial Not a Number
NEGATIVE_INFINITY	Representa el infinito negativo
POSITIVE_INFINITY	Representa el infinito positivo

```
alert (Number.MAX_VALUE);
alert (Number.MIN_VALUE);
alert (Number.NEGATIVE_INFINITY);
alert (Number.POSITIVE_INFINITY);
```

Métodos

- **toExponential ()** Convierte el número en una notación exponencial
- **toFixed()** Formatea el número con la cantidad de dígitos decimales

que pasemos por parámetro.

- **toPrecision()** Formatea el número con la longitud que pasemos como parámetro.

```
var n=100000000;
alert(n.toExponential());  // 1e+8
n=3.1416;
alert(n.toFixed(2));      // 3.14
n=50000.986;
alert(n.toPrecision(6));   // 50001.0
alert(n.toPrecision(2));   // 5.0e+4
```