

TEMA 5 (Parte B). Objetos.

Desarrollo Web en Entorno Cliente.

Profesor: Juan José Gallego García

Índice :

- Objetos. Propiedades y Métodos.
- Métodos Get y Set.
- Objetos predefinidos. Math, Date ...
- Bibliografía.

Objetos.

En JS toma gran importancia los objetos, ya que la mayoría de los elementos son objetos. Incluso los tipos primitivos podrían declararse como objetos con la palabra clave “new” Ej: `var x = new Number(5);`. Los objetos son una colección de valores con nombres, algo similar a los arrays asociativos en otros lenguajes. Poseen propiedades. Son tratados por referencia, no por valor.

- Para crear un objeto podemos usar varias formas: mediante `Object.create()` basado en otro objeto, usando la palabra clave “new” y definiendo las propiedades, o de la forma más directa y eficiente usando un **objeto literal** definiéndolo entre `{}`.

```
var alumno = {nombre: "Juan", apellido: "Pérez", edad : 23};  
alert ("Nombre : " + alumno.nombre); // " Nombre : Juan "
```

```
var estudiante=Object.create(alumno);
```

```
var alumno = new Object ( );  
alumno.nombre = "Juan";  
alumno.apellido= "Pérez";  
alumno.edad = 23;  
alert ("Nombre : " + alumno.nombre);
```

- La forma de objeto literal también puede ser declarada como sigue :

```
var alumno = {  
  nombre: "Juan",  
  apellido: "Pérez" ,  
  edad : 23  
};  
alert ("Nombre : " + alumno.nombre);
```

```
var alumno = {};  
alumno.nombre = "Juan";  
alumno.apellido = "Pérez";  
alumno.edad = 23;  
alert ("Nombre : " + alumno.nombre);
```

Propiedades.

Las propiedades son los pares de nombre + valor asociados a los objetos. Éstas pueden ser creadas, modificadas y eliminadas. En el ejemplo anterior **nombre : "Juan"** sería una propiedad del objeto **alumno**.

- Para acceder a una propiedad podemos usar las siguientes sintaxis :

nombreObjeto.nombreProp , **nombreObjeto ["nombreProp"]** ó

nombreObjeto [x] , donde **x** es una variable (**nombreProp**)

- Podemos recorrer todas las propiedades de un objeto con **for/in**

```
var alumno = {  
  nombre: "Juan",  
  apellido: "Pérez" ,  
  edad : 23  
};  
  
for (prop in alumno) {  
  alert (prop + ": " + alumno[prop]);  
}
```

- Podemos agregar una nueva propiedad al objeto y borrarla con **"delete"**:

```
alumno.curso = "2º";  
delete alumno.curso;
```

Métodos.

Los métodos son acciones, mediante funciones, que se pueden aplicar a los objetos y que se almacenan como propiedades (nombre + definición función).

Ej:

```
var alumno = {  
  nombre: "Juan",  
  apellido: "Pérez" ,  
  edad : 23,  
  muestra: function () {alert("Nombre: "+this.nombre + "\nApellido: "+this.apellido); }  
};  
alumno.muestra();
```

La palabra clave **this** hace referencia al objeto propietario, en este caso **alumno**.

En el ejemplo anterior el método **muestra** accede a las propiedades **nombre** y **apellidos** .

Al igual que con las demás propiedades podemos añadir un nuevo método al objeto.

```
var alumno = {  
    nombre: "Juan",  
    apellido: "Pérez" ,  
    edad : 23  
};  
  
alumno.muestra =function () {alert("Nombre: "+this.nombre + "\nApellido: "+this.apellido); }  
alumno.muestra();
```

Métodos Get y Set. Como métodos especiales para abreviar la lectura y establecer el valor de una propiedad se utilizan los métodos **get** y **set**. El tratamiento es como si fueran propiedades.

[Ver otros métodos de objetos ...](#)

```
var alumno = {  
    nombre: "Juan",  
    edad : 23 ,  
    set ponerNombre(nombre1) {this.nombre=nombre1},  
    get leerNombre( ) {alert(this.nombre)}  
};  
  
alumno.leerNombre; // Muestra "Juan"  
alumno.ponerNombre="Pepe";  
alumno.leerNombre; // Muestra "Pepe"
```

Como resumen de declaración de nuevos objetos de distintos tipos tenemos:

```
var x1 = new Object();    // Nuevo objeto de tipo objeto.
var x2 = new String();    // Nuevo objeto de tipo cadena.
var x3 = new Number();    // Nuevo objeto numérico.
var x4 = new Boolean();   // Nuevo objeto booleano.
var x5 = new Array();     // Nuevo objeto array.
var x6 = new RegExp();    // Nuevo objeto tipo expresión regular.
var x7 = new Function();  // Nuevo objeto función.
var x8 = new Date();      // Nuevo objeto fecha.
```

Su equivalente literal, **mucho más eficiente**:

Es bastante útil el uso de `typeof` para saber qué tipo de dato almacena una variable.

```
var a=1;
alert (typeof a);
//Devuelve 'number'
```

```
{}
```

 en lugar de `new Object()`.
`""` en lugar de `new String()`.
`123` en lugar de `new Number()`.
`true` / `false` en lugar de `new Boolean()`.
`[]` en lugar de `new Array()`.
`/()/` en lugar de `new RegExp()`.
`() {}` en lugar de `new Function()`.

Objetos predefinidos.

Existen varios objetos basados en un prototipo que poseen propiedades predefinidas y las cuales podemos usar:

Math.

Propiedades:

```
Math.E      // devuelve el número de Euler
Math.PI     // devuelve PI
Math.SQRT2  // devuelve la raíz cuadrada de 2
Math.SQRT1_2 // devuelve la raíz cuadrada de 1/2
Math.LN2    // devuelve logaritmo neperiano de 2
Math.LN10   // devuelve logaritmo neperiano de 10
Math.LOG2E  // devuelve el logaritmo de E base 2
Math.LOG10E // devuelve el logaritmo de E base 10
```

Métodos (Sintaxis: `Math.método`)

```
abs(x) // devuelve valor absoluto de x.
acos(x) // devuelve el arcocoseno de x, en radianes.
asin(x) // devuelve arcoseno de x, en radianes.
atan(x) // devuelve arcotangente de x, valor entre  $-\pi/2$  y  $\pi/2$  radianes.
atan2(y, x) // devuelve arcotangente del cociente de sus argumentos.
ceil(x) // devuelve el entero más próximo por arriba.
cos(x) // devuelve el coseno de x (x is in radians)
exp(x) // devuelve el valor de E elevado a x.
floor(x) // devuelve el entero más próximo por debajo.
log(x) // devuelve el logaritmo neperiano de x.
max(x, y, z, ..., n) // devuelve el valor más alto.
min(x, y, z, ..., n) // devuelve el valor más bajo.
pow(x, y) // devuelve el valor del primer parámetro elevado al segundo
random() // devuelve un valor aleatorio entre 0 y 1
round(x) // devuelve el valor de un número redondeado a su número entero más próximo.
sin(x) // devuelve el seno de x (x en radianes)
sqrt(x) // devuelve la raíz cuadrada de x.
tan(x) // devuelve la tangente de un ángulo.
```

Ejemplos:

```
Math.PI;           // devuelve 3.141592653589793
Math.round(4.7);   //devuelve el valor de un número redondeado a su número entero más próximo: Devuelve 5
Math.pow(2,3);      // devuelve el valor del primer parámetro elevado al segundo: Devuelve 8
Math.sqrt(16);      // Devuelve la raíz cuadrada de un número: Devuelve 4
Math.abs(-3);       // Devuelve el valor absoluto de un número: Devuelve 3
Math.ceil(4.4);     // Devuelve el entero más próximo por arriba: Devuelve 5
Math.floor(4.7);    // Devuelve el entero más próximo por debajo: Devuelve 4
Math.sin(3.1416);   // Devuelve el seno (un valor entre -1 y 1) del ángulo (dada en radianes): Devuelve ~ 0
```

Date.

Para usar un objeto de fecha debemos declararlo de alguna de las siguientes formas:

```
new Date() Hora actual, formato estándar.
new Date(año,mes,dia,horas,minutos,segundos,milisegundos)
new Date(milisegundos)
new Date(cadena fecha)
```

También se acepta **Date()** directamente para devolver la fecha formato estándar.

Ejemplos:

```
var d1= new Date ();  
var d2 = new Date(2019, 10, 04, 13, 0, 0, 0);  
var d3 = new Date(1560000000000);  
var d4 = new Date("November 04, 2019 13:00:00");  
alert (d1); // devuelve la fecha y hora actual en formato estándar con referencia GMT.  
alert (d2); // devuelve la hora establecida por los parámetros.  
alert (d3); // devuelve la hora establecida por cadena formato milisegundos (aprox. 15:20, 8 junio 2019)  
            // '0' milisegundos sería la fecha base: 1 de enero de 1970 a las 00:00:00 UTC  
alert (d4); // devuelve la hora establecida por cadena formato fecha.  
var d = new Date("06-01-2019"); // Formato corto para establecer una fecha. 1 junio 2019.
```

Aunque existen diversos métodos para manejar las fechas y horas nos quedaremos con los más prácticos. Por ejemplo al igual que para GMT existen métodos equivalente para el sistema universal UTC. [\(Ver sistemas de tiempo\)](#)

Métodos.

```
getFullYear() // Devuelve el año con 4 dígitos (yyyy)
getMonth()    // Devuelve el mes (0-11)
getDate()     // Devuelve el día del mes (1-31)
getHours()    // Devuelve la hora (0-23)
getMinutes()  // Devuelve los minutos (0-59)
getSeconds()  // Devuelve los segundos (0-59)
getMilliseconds() // Devuelve los milisegundos (0-999)
getTime()     // Devuelve los milisegundos desde 1 de Enero de 1970
getDay()      // Devuelve el día de la semana (0-6)
Date.now()    // Devuelve los milisegundos desde tiempo base. ECMAScript 5.
```

```
setDate() // Establece el día (1-31)
setFullYear() // Establece el año.
setHours() // Establece la hora (0-23)
setMilliseconds() // Establece los milisegundos (0-999)
setMinutes() // Establece los minutos (0-59)
setMonth() // Establece el mes (0-11)
setSeconds() // Establece los segundos (0-59)
```

Ejemplos :

```
var d = new Date(); // Crea objeto fecha y devuelve la fecha actual.  
alert (d.getDate()+'\/'+d.getMonth()+'\/'+d.getFullYear()); // De la fecha anterior componemos DD/MM/AAAA.  
alert (d.getHours()+':'+d.getMinutes()); // Componemos HH:MM
```

El siguiente ejemplo calcula en segundos el tiempo que se tarda en pulsar aceptar en la primera ventana de alerta.

```
var t1=Date.now();  
alert ("Espera ...");  
var t2=Date.now();  
var t=t2-t1;  
alert("Tiempo de espera "+t/1000+ " segundos");
```

Otro ejemplo:

```
var d = new Date(); // Crea objeto fecha actual.  
alert (d.getFullYear()); // Devuelve el año actual.  
d.setFullYear(2018); // Cambiamos el año en el objeto fecha.  
alert (d.getFullYear()); // Devuelve 2018.
```

Bibliografía

- LibrosWeb
- W3Schools
- Developer Mozilla Docs