TEMA 5 (Parte A). Numbers, Strings, Arrays.

Desarrollo Web en Entorno Cliente.

Profesor: Juan José Gallego García

Índice :

Numbers.

Hasta ahora hemos usado variables de tipo numérica, pero debemos conocer algunas características de estos valores en JS, así como ampliar su uso con las propiedades y métodos predefinidos que poseen.

- JS posee sólo un tipo de número con o sin decimales.
- Todos los números se almacenan como números de coma flotante de doble precisión 64 bits.
- Se pueden expresar en notación científica Ej: `var x = 123e5;`
- Tienen una precisión de 15 dígitos: `var x = 999999999999999;`
- El número máximo de decimales que podemos usar en un decimal es 17.

- NaN es una palabra reservada en JS para indicar que una operación da como resultado un valor no numérico. Ej:

```
var x= 3;
alert (x/"hola"); // Devuelve NaN
```

- Podemos usar Infinity o -Infinity para operar o comparar en representación de los números mayor y menor posible que puede manejar JS.

```
var x= 20000000000;
if (x < Infinity) alert ("Ok"); // Devuelve OK.
```

- Para declarar variables en hexadecimal podemos usar “0x”. toString (base) cambia el valor en la base numérica necesaria. Base máxima: “36”.

```
var x= 0xFF; // x tomará el valor 255
var x = 32;
x.toString(10);  // devuelve 32
x.toString(32);  // devuelve 10
x.toString(16);  // devuelve 20
x.toString(8);   // devuelve 40
x.toString(2);   // devuelve 100000
```

Métodos y propiedades de números (Numbers)

En JS los tipos primitivos poseen métodos y propiedades como los objetos.

- El método toString( ) convierte un número en una cadena

```
var x = 7 ;
x.toString(); //Devuelve "7"
```

- El método toExponential ( ) devuelve un número redondeado y en notación exponencial.

```
var x = 9.656;
x.toExponential(2);     // devuelve 9.66e+0
x.toExponential(4);     // devuelve 9.6560e+0
x.toExponential(6);     // devuelve 9.656000e+0
```

- toFixed ( ) redondea un número con un determinado números de decimales.

```
var x = 9.656;
x.toFixed(0);           // devuelve 10
x.toFixed(2);           // devuelve 9.66
x.toFixed(4);           // devuelve 9.6560
x.toFixed(6);           // devuelve 9.656000
```

- Los métodos Number( ), parseFloat( ), parseInt( ) convierten objetos y cadenas en valores numéricos. El primero para cualquier tipo de dato u objeto, y los dos últimos para decimales y enteros respectivamente.

```
Number(true);           // devuelve 1
Number("10.33");        // devuelve 10.33
parseInt("10.33");       // devuelve 10
parseFloat("10.33");      // devuelve 10.33
```

- Number.MAX_VALUE Devuelve el mayor número posible en JS.
  - Para enteros mayores al máximo posible en JS usar BigInt (usado en cálculos financieros).
- Number.MIN_VALUE Devuelve el menor número posible en JS
- Number.POSITIVE_INFINITY representa el infinito positivo.
- Number.NEGATIVE_INFINITY representa el infinito negativo.

```
alert (Number.MAX_VALUE) ; // Devuelve 1.7976931348623157e+308
```

Strings.

Las cadenas de caracteres strings pueden almacenar y manipular textos.

- Podemos usar comillas simples o dobles para declarar una cadena.

```
var x = "hola";
var x = 'hola';
```

- Si queremos mostrar comillas en una expresión podemos alternar entre los dos tipos de comillas.

```
var x = " 'hola' ";
alert (x); // Muestra  'hola'
```

- Para acortar una cadena y dividirla en varias líneas podemos usar ‘ \ ’. Tras la barra no debe de haber ningún espacio. También podemos usar el operador “ + “ , y más eficiente el carácter especial “ ` “ (tilde invertida)

```
var x = "hola hola hola " +
"hola hola";
```

```
var x = "hola hola hola \
hola hola";
```

```
var x = ` Varios
          saltos de
          líneas `;
```

```
var y = 3;
var x = ` ${y}
          saltos de
          líneas `;
//Despliega valor de 'y'
```

Métodos y propiedades de cadenas (Strings).

```
var x = "hola hola";
alert (x.length);
//Devuelve 9
```

- La propiedad length devuelve la longitud de una cadena.
- El método search ( ) busca una cadena y devuelve la posición de la primera coincidencia comenzando desde 0. Ver también indexOf( ) y lastIndexOf( )

```
var x = "hola hola";
alert (x.search ("la")); // Devuelve 2
```

- El método slice ( ) extrae una parte de una cadena y la guarda en otra cadena. Posee dos parámetros, posición inicial (comenzando en 0) y posición final +1 de la cadena que queremos extraer. Si los parámetros son negativos se comienza desde el final de la cadena.

```
var x = "hola que tal";
var y= x.slice (5,8);
alert (y); Muestra “que”
```

- El método replace ( ) reemplaza una cadena por otra. De esta forma sólo cambia la primera coincidencia y distingue entre mayúsculas y minúsculas. Para encontrar mayúsculas y minúsculas usamos en lugar de “cero” → **/cero/i** , y para encontrar todas las coincidencias **/cero/g** (Ver objeto RegExp)

```
var x = "Todos los alumnos sacarán un cero.";
var y = x.replace ("cero","diez");
alert (y); // Todos los alumnos sacarán un diez.
```

- toUpperCase( ) y t

```
var x = "Hola que tal.";
alert (x.toUpperCase()); // Muestra “HOLA QUE TAL.”
```

úsculas y minúsculas respectivamente.
- charAt (posición) y charCodeAt(posición) , extrae un carácter de una cadena y devuelve el unicode (UTF-16) respectivamente indicado en “posición”.

```
var x = "Hola que tal.";
alert (x.charAt(0)); // Muestra “H”
```

```
var x = "Hola que tal.";
alert (x.charCodeAt(0));// Muestra 72
```

- Podemos acceder a un carácter de la cadena a través de un índice directamente (como array).

```
var x = "Hola que tal.";
alert ( x [1] ); // Devuelve "o"
```

- split (" , ") , convierte una cadena en un array usando un carácter para separar los elementos. En el siguiente ejemplo en lugar de usar "," podemos usar un espacio para separar la palabras.

```
var x = "Hola que tal.";
var y= x.split (" ");
alert ( y [1] ); // Devuelve "que"
```

- Para crear un array con las letras de una palabra usamos comillas juntas.

```
var x = "Hola";
var y= x.split ("");
alert ( y [1] ); // Devuelve 'o'
```

Arrays.

Los arrays se usan para almacenar múltiples valores (del mismo o distinto tipo) en una sola variable. Esto permite recorrer (por su índice) fácilmente esos valores, por ejemplo, a través de bucles.

- La sintaxis para crear un array : var nombreVar = [valor 1, valor 2 ….]

```
var x = [ 1 , 2 , 3 ];
alert (x [1] ); //Devuelve 2
```

```
var x = [ "uno" , "dos" , 3 ];
alert (x [1] ); // Devuelve "dos"
```

```
var x = [ "uno" , "dos" ,"tres" ];
alert ( x );
// Devuelve " uno , dos , tres "
```

- Para establecer un valor a un elemento del array. Ej :

```
var x = [ "uno" , "dos" , 3 ];
alert ( x [ 2 ] ); // Devuelve 3
x [ 2 ] = "tres" ;
alert ( x [ 2 ] ); // Devuelve "tres"
```

Métodos y propiedades de los arrays.

- La propiedad length devuelve el número de elementos de un array.

```
var x = [ "uno" , "dos" , 3 , 1 ];
alert ( x.length ); // Devuelve 4
```

- Recorrido del array.
  - Mediante bucle for
  - Mediante método foreach ( )

```
<body>
<p id="p1"></p>
<script>
  var p1= document.getElementById("p1");
   var x = [ "uno" , "dos" , 3 , 1 ];
    for ( i=0; i< x.length; i ++) {
      p1.innerHTML += x [i] + "<br>";
    }
</script>
</body>
```

```
<body>
<p id="p1"></p>
<script>
  var p1= document.getElementById("p1");
  var x = [ "uno" , "dos" , 3 , 1 ];
  x.forEach(f1);
  function f1 (elemento) {
    p1.innerHTML += elemento + "<br>";
  }
</script>
</body>
```

- Los métodos toString ( ) y join ( ) convierten todos los elementos de un array a una cadena. El primero separados con “,” el segundo se puede elegir el separador.

```
var x = [2,3,4,5];
var cadena = x.toString ();
alert ( cadena ); // Devuelve 2,3,4,5
cadena = x.join ("-") ;
alert ( cadena ) ; // Devuelve 2-3-4-5
```

- Declaramos un array vacío (dinámico), podemos añadir un nuevo elemento al final del array con push ( ) , o asignar los valores mediante índices.

```
var x = [ ]; // Declara array vacío.
x.push ("alumno1");
x.push ("alumno2");
alert ( x ) ; //Devuelve " alumno1 , alumno2 "
```

```
var x = [ ]; // Declara array vacío.
x[0]= "alumno1";
x[1]= "alumno2";
alert ( x ) ; // Devuelve " alumno1 , alumno2 "
```

- Para eliminar el elemento final de un array usaremos pop ( ). Devuelve el elemento que se eliminó.

```
var x = [2,3,4,5];
alert (x.pop ()); // Muestra 5
alert ( x ) ; // Muestra 2,3,4
```

- El método unshift( ) agrega un nuevo elemento al principio del array, y desplaza los elementos originales. Devuelve la nueva longitud del array.

```
var x = [2,3,4,5];
alert (x.unshift (1)); // Muestra 5
alert ( x ) ; // Muestra 1,2,3,4,5
```

- El método shift( ) elimina el primer elemento del array y "desplaza" todos los demás elementos a un índice inferior. Devuelve el elemento que se eliminó.

```
var x = [2,3,4,5];
alert (x.shift ()); // Muestra 2
alert ( x ) ; // Muestra 3,4,5
```

- El método splice ( ) añade nuevos elementos al array pudiéndose borrar algunos o ninguno de los originales a través de los parámetros.

```
var x = [2,3,4,5];
x.splice ( 2, 1 , 0.1 , 0.2 );
alert ( x ) ; // Muestra 2,3,0.1,0.2,5
```

- El método concat ( ) crea un nuevo array de la concatenación de dos o más arrays.

```
var x = [2,3,4,5];
var y = [6,7,8,9];
var z = [10,11,12,13];
var total = x.concat( y,z );
alert ( total ) ; // Muestra 2,3,4,5,6,7,8,9,10,11,12,13
```

- El método slice ( ) extrae un nuevo array de otro original. Se puede especificar el intervalo de elementos extraídos a través de 2 parámetros, inicial y final (no incluido). Si se omite el segundo parámetro, extrae los elementos desde el elemento especificado por el primer parámetro hasta el último.

```
var x = ["uno","dos","tres","cuatro"];
var y=x.slice(1,3);
alert ( x + "\n"+ y ) ; // Muestra "uno,dos,tres,cuatro "
                        //          "dos,tres"
```

- El método reverse ( ) invierte el orden de un array.

```
var x = ["uno","dos","tres","cuatro"];
alert ( x.reverse()) ;  // Muestra "cuatro,tres,dos,uno"
```

- El método sort ( ) ordena alfabéticamente un array.

```
var x = ["uno","dos","tres","cuatro"];
alert ( x.sort()) ; // Muestra "cuatro,dos,tres,uno"
```

- El método anterior ordena cadenas de caracteres, por ello, “2” sería mayor a “10” .Para ordenar números es necesario usar una función de comparación.

```
var x = [1,4,3,2,10,7];
alert ( x.sort(function (a,b){return a-b})) ; Devuelve 1,2,3,4,7,10
```

Ordena ascendentemente, si usamos “**return b-a** “ , ordenará descendentemente.

- Para calcular el número mayor y menor en un array podemos usar el código,

```
  var x = [1,-4,3,2,10,7];
  x.sort(function (a,b){return a-b}) ;
  var min=x[0];
  var max=x[x.length-1];
  alert ("Mayor : "+max + " Menor : "+min); Muestra “Mayor
: 10 Menor : - 4 “
```

ó con el objeto/clase **Math**:

```
  var x = [1,-4,3,2,10,7];
  alert (Math.max (...x));
  alert(Math.min (...x));
```

- El método map ( ) crea un nuevo array aplicando una función determinada a cada uno de sus elementos. Dicha función recibe 3 parámetros, el **valor** del elemento , el **índice** y el propio **array**. Se puede omitir los parámetros índice y array.

Calcula el cuadrado de cada elemento.

```
var x = [1,-4,3,2,10,7];
var y = x.map (f1);
function f1 (valor, index, array){
  return valor * valor ;
}
alert (y) ; // Muestra 1,16,9,4,100,49
```

```
var x = [1,-4,3,2,10,7];
var y = x.filter (f1);
function f1 (valor){
  return valor > 2 ;
}
alert (y) ; // Muestra 3,10,7
```

- El método filter ( ) crea un nuevo array con los elementos que cumplen una determinada condición en la función.
- Formas de copiar un array para no tocar el original. Ej :

```
var x = [1,-4,3,2,10,7];
var y = [].concat(x); // Primera forma
var y = [...x];  // Segunda forma
x[0]=0;
alert (x) ;
alert (y) ;
```

- Ver otros métodos : reduce ( ), every ( ) , some ( ), find ( ), findIndex(), indexOf(), ...

Arrays multidimensionales.

Al igual que los arrays de una dimensión podemos declarar arrays de dos o más dimensiones. En este caso hay que tener cuidado con los límites durante el recorrido de los elementos para que no nos de un error fuera de rango. La sintaxis es la misma, excepto que en lugar de ser elementos simples, cada elemento es otro array.

En el siguiente ejemplo se declara un array de dos dimensiones compuesto de 3 elementos, y cada uno de ellos a su vez de arrays de 4 elementos. (3x4)

```
var x = [[1,2,3,4],[5,6,7,8],[9,10,11,12]];
for (i = 0 ; i < x.length ; i++) {
   for (j = 0 ; j < x[i].length ; j++) {
      alert ("Elemento : "+"["+i+"]"+"["+j+"] = "+ x[i][j]);
   }
}
```

En este otro ejemplo declaramos un array 3x2x2

```
var x = [ [ [1,2],[3,4] ] , [ [5,6],[7,8] ] , [ [9,10],[11,12] ] ];
for (i = 0 ; i < x.length ; i++) {
   for (j = 0 ; j < x[i].length ; j++) {
       for (t = 0 ; t < x[i][j].length ; t++) {
          alert ("Elemento : "+"["+i+"]"+"["+j+"]"+"["+t+"] = "+x[i][j][t]);
       }
    }
}
```

Al igual que hicimos con los arrays de una dimensión, podríamos declarar los arrays multidimensionales dinámicamente haciendo uso de push ( ) para añadir elementos, ó mediante asignación.

```
var x = [];
var cont = 0;
for (i = 0 ; i <= 3 ; i++) {
   x.push([]);
   for (j = 0 ; j <= 4 ; j++) {
      x[i].push(cont ++);
    }
    alert (x[i]);
}
```

<- equivalente ->

```
var x = [];
var cont = 0;
for (i = 0 ; i <= 3 ; i++) {
      x[i]=[];
      for (j = 0 ; j <= 4 ; j++) {
            x[i][j]= cont++;
       }
      alert (x[i]);
}
```

Bibliografía

- LibrosWeb
- W3Schools
- Developer Mozilla Docs