MODELO DE OBJETOS PREDIFINIDOS EN JAVASCRIPT

Unidad didáctica 4 Parte 2

Modelo de Objetos Predefinidos en JavaScript

Objeto String

- Una cadena (string) consta de uno o más caracteres de texto, rodeados de comillas simples o dobles; da igual cuáles usemos ya que se considerará una cadena de todas formas, pero en algunos casos resulta más cómodo el uso de unas u otras.
  - Uso de comillas simples
  - Uso de comillas dobles
  - Concatenación con el operador +

Objeto String (Caracteres de escape)

Caracteres de escape	
Caracteres	**Descripción**
`\"`	Comillas dobles
`\'`	Comilla simple
`\\`	Barra invertida
`\b`	Retroceso
`\t`	Tabulador
`\n`	Nueva línea
`\r`	Salto de línea
`\f`	Avance de página

Objeto String (Propiedades)

- Para crear un objeto String lo podremos hacer de dos formas:

```
var miCadena = new String("texto de la cadena");
var miCadena = "texto de la cadena";
```

```
cadena.propiedad;
cadena.metodo( [parámetros] );
```

Propiedades del objeto String	
Propiedad	**Descripción**
`length`	Contiene la longitud de la cadena de texto.

Objeto String (Métodos)

Métodos del objeto String	
Método	**Descripción**
`charAt()`	Devuelve el carácter de la posición indicada entre paréntesis.
`charCodeAt()`	Devuelve el Unicode del carácter de la posición indicada entre paréntesis.
`concat()`	Devuelve el resultado de la unión (concatenación) de una o más cadenas.
`endsWith()`	Cheque si una cadena termina de una forma concreta.
`fromCharCode()`	Convierte valores Unicode a caracteres.
`includes()`	Chequea si una cadena contiene una subcadena concreta.
`indexOf()`	Devuelve la posición donde encuentra por primera vez el carácter buscado en la cadena.
`lastIndexOf()`	Devuelve la posición donde encuentra por última vez el carácter buscado en la cadena.
`localeCompare()`	a.localeCompare(b) Compara dos cadenas a y b devolviendo: • -1 si la cadena a es menor que b. • 0 si son iguales. • 1 si la cadena a es mayor que b.
`match()`	Devuelve las coincidencias encontradas entre una expresión regular y una cadena o null si no encuentra ninguna coincidencia.
`repeat()`	Devuelve una cadena con el resultado de la repetición veces de la cadena original.
`replace()`	Reemplaza una subcadena con una nueva cadena.
`search()`	Devuelve la posición dónde encontró una subcadena o la subcadena que concuerda con una expresión regular.
`slice()`	Devuelve una parte de la cadena.
`split()`	Divide una cadena en un array de subcadenas.
`startsWith()`	Chequea si una cadena comienza por algo concreto.
`substr()`	Devuelve un número determinado de caracteres comenzando en una determinada posición.
`substring()`	Extrae los caracteres de una cadena entre dos posiciones concretas.
`toLowerCase()`	Devuelve la cadena en minúsculas.
`toString()`	Devuelve el valor de un objeto String.
`toUpperCase()`	Devuelve la cadena en mayúsculas.
`trim()`	Quita los espacios sobrantes al principio y al final de una cadena.
`valueOf()`	Devuelve el valor primitivo de una cadena.

Objeto String (Propiedades de presentación)

Métodos de presentación del objeto String	
Método	**Descripción**
`anchor("nombre")`	Crea una etiqueta ancla `<a name="nombre">cadena</a>` donde `cadena` es el objeto `String` que llama al método `anchor`.
`big()`	Visualiza la cadena utilizando una fuente grande.
`blink()`	Visualiza la cadena parpadeando.
`bold()`	Visualiza la cadena en negrita.
`fixed()`	Visualiza la cadena utilizando una fuente regular.
`fontcolor("color")`	Visualiza la cadena utilizando un color concreto.
`fontsize(tamaño)`	Visualiza la cadena utilizando un tamaño concreto.
`italics()`	Visualiza la cadena en cursiva.
`link("url")`	Visualiza la cadena creando un hiperenlace.
`small()`	Visualiza la cadena utilizando una fuente pequeña.
`strike()`	Visualiza la cadena tachada.
`sub()`	Visualiza la cadena como subíndice.
`sup()`	Visualiza la cadena como superíndice.

Objeto Math

- Disponemos de un objeto **Math** en JavaScript, que nos permite realizar operaciones matemáticas. El objeto **Math** no es un constructor (no nos permitirá por lo tanto crear o instanciar nuevos objetos que sean de tipo Math), por lo que, para llamar a sus propiedades y métodos, lo haremos anteponiendo Math a la propiedad o el método. Por ejemplo.
  - var x = Math.PI; // Devuelve el número PI
  - var y = Math.sqrt(16); // Devuelve la raíz cuadrada de 16.

Objeto Math (Propiedades)

Propiedades del objeto Math	
Propiedad	**Descripción (son constantes)**
`E`	Devuelve el número Euler o constante de Napier (aproximadamente 2,718).
`LN2`	Devuelve el logaritmo neperiano de 2 (aproximadamente 0,693).
`LN10`	Devuelve el logaritmo neperiano de 10 (aproximadamente 2,302).
`LOG2E`	Devuelve el logaritmo en base 2 del número E (aproximadamente 1,442).
`LOG10E`	Devuelve el logaritmo en base 10 del número E (aproximadamente 0,434).
`PI`	Devuelve el número PI (aproximadamente 3,14159).
`SQRT1_2`	Devuelve la raíz cuadrada de ½ (aproximadamente 0,707106).
`SQRT2`	Devuelve la raíz cuadrada de 2 (aproximadamente 1,41421).

Objeto Math (Métodos)

Métodos del objeto Math	
Método	**Descripción**
`abs(x)`	Devuelve el valor absoluto de x.
`acos(x)`	Devuelve el arcocoseno de x, en radianes.
`asin(x)`	Devuelve el arcoseno de x, en radianes.
`atan(x)`	Devuelve el arcotangente de x, en radianes con un valor entre -PI/2 y PI/2.
`atan2(y,x)`	Devuelve el arcotangente del cociente de sus argumentos.
`ceil(x)`	Devuelve el número x redondeado al alza al entero más próximo.
`cos(x)`	Devuelve el coseno de x (x está en radianes).
`exp(x)`	Devuelve E elevado a x.
`floor(x)`	Devuelve el número x redondeado a la baja al entero más próximo.
`log(x)`	Devuelve el logaritmo neperiano (en base E) de x.
`max(x,y,z,...,n)`	Devuelve el número más alto de los que se pasan como parámetros.
`min(x,y,z,...,n)`	Devuelve el número más bajo de los que se pasan como parámetros.
`pow(x,y)`	Devuelve el resultado de x elevado a y.
`random()`	Devuelve un número al azar entre 0 y 1.
`round(x)`	Redondea x al entero más próximo.
`sin(x)`	Devuelve el seno de x (x está en radianes).
`sqrt(x)`	Devuelve la raíz cuadrada de x.
`tan(x)`	Devuelve la tangente de un ángulo.
`trunc(x)`	Devuelve la parte entera de x.

Objeto Number

- El objeto **Number** se usa muy raramente, ya que, para la mayor parte de los casos, JavaScript satisface las necesidades del día a día con los valores numéricos que almacenamos en variables. Pero el objeto Number contiene alguna información y capacidades muy interesantes para programadores más serios.

Objeto Number (Propiedades)

Propiedades del objeto Number	
Propiedad	**Descripción**
`constructor`	Devuelve la función que creó el objeto `Number`.
`MAX_VALUE`	Devuelve el número más alto disponible en JavaScript.
`MIN_VALUE`	Devuelve el número más pequeño disponible en JavaScript.
`NEGATIVE_INFINITY`	Representa a infinito negativo (se devuelve en caso de overflow).
`NaN`	Representa un valor "Not a Number"
`POSITIVE_INFINITY`	Representa a infinito positivo (se devuelve en caso de overflow).
`prototype`	Permite añadir nuestras propias propiedades y métodos a un objeto.

Objeto Number (Métodos)

Métodos del objeto Number	
Método	**Descripción**
`isFinite()`	Chequea si el número es un valor finito.
`isInteger()`	Chequea si es un valor entero.
`isNaN()`	Chequea si es un "Not a Number".
`isSafeInteger()`	Chequea si un número es un entero en el rango $[-(2^{53} - 1) , (2^{53} -1)]$.
`toExponential(x)`	Convierte un número a su notación exponencial.
`toFixed(x)`	Formatea un número con `x` digitos decimales después del punto decimal.
`toPrecision(x)`	Formatea un número a `x` dígitos de longitud. (Sin incluir el punto decimal)
`toString(x)`	Convierte un objeto `Number` en una cadena. (x debe ser un número comprendido entre 2 y 36, será la base a la que se quiere cambiar el nº). • Si x=2 se mostrará el número en binario. • Si x=8 se mostrará el número en octal. • Si x=16 se mostrará el número en hexadecimal. • Si x se omite mostrará el mismo número.
`valueOf()`	Devuelve el valor primitivo de un objeto `Number`.

Objeto Boolean (Definición, Propiedades y Métodos)

- El objeto **Boolean** se utiliza para convertir un valor no Booleano a un valor Booleano (true o false).

Propiedades del objeto Boolean	
Propiedad	**Descripción**
`constructor`	Devuelve la función que creó el objeto `Boolean`.
`prototype`	Permite añadir nuestras propias propiedades y métodos a un objeto.

Métodos del objeto Boolean	
Método	**Descripción**
`toString(x)`	Convierte un valor `Boolean` en una cadena y devuelve el resultado (`true` o `false`).
`valueOf()`	Devuelve el valor primitivo de un objeto `Boolean`.

Objeto Date

- El objeto **Date** se utiliza para trabajar con fechas y horas.
- Una fecha en Java Script se puede escribir:
  - Sat Dec 09 2017 10:05:06 GMT+0100
  - 1512810306191
- Cuando se escribe en número representa el número de milisegundos que han pasado desde el 1 de enero de 1970 a las 00:00:00.
- Un día tiene 24 horas, un total de 86.400.000 milisegundos (24x60x60x1000).
- Los objetos Date se crean con new Date().
  - var d = new Date();
  - var d = new Date(1512810306191);
  - var d = new Date("October 13, 2014 11:13:00");
  - var d = new Date(2017,11,9,10,20,0,0);
  - var d = new Date(2017,11,9);

Objeto Date (Propiedades)

Propiedades del objeto Date	
Propiedad	**Descripción**
`constructor`	Devuelve la función que creó el objeto `Date`.
`prototype`	Permite añadir nuestras propias propiedades y métodos a un objeto.

Objeto Date (Métodos)

Métodos del objeto Date	
Método	**Descripción**
`getDate()` `setDate()`	Devuelve o establece el día del mes (de 1-31).
`getDay()`	Devuelve el día de la semana (de 0-6). (0-Domingo,….6-Sábado)
`getFullYear()` `setFullYear()`	Devuelve o establece el año (4 dígitos).
`getHours()` `setHours()`	Devuelve o establece la hora (de 0-23).
`getMilliseconds()` `setMilliseconds()`	Devuelve o establece los milisegundos (de 0-999).
`getMinutes()` `setMinutes()`	Devuelve o establece los minutos (de 0-59).
`getMonth()` `setMonth()`	Devuelve o establece el mes (de 0-11).
`getSeconds()` `setSeconds()`	Devuelve o establece los segundos (de 0-59).
`getTime()` `setTime()`	Devuelve o establece los milisegundos desde media noche del 1 de Enero de 1970.
`getTimezoneOffset()`	Devuelve la diferencia de tiempo entre GMT y la hora local en minutos.
`getUTCDate()` `setUTCDate()`	Devuelve o establece el día del mes en base a la hora UTC (de 1-31).
`getUTCDay()`	Devuelve el día de la semana en base a la hora UTC (de 0-6).
`getUTCFullYear()` `setUTCFullYear()`	Devuelve o establece el año en base a la hora UTC (4 dígitos).
`getUTCHours()` `setUTCHours()`	Como getHours() pero hora UTC.

Objeto Date (Métodos)

Métodos del objeto Date	
Método	**Descripción**
`getUTCMilliseconds()` `setUTCMilliseconds()`	Como getMilliseconds() pero milisegundos UTC.
`getUTCMinutes()` `setUTCMinutes()`	Como getMinutes() pero minutos UTC.
`getUTCMonth()` `setUTCMonth()`	Como getMonth() pero mes UTC.
`getUTCSeconds()` `setUTCSeconds()`	Como getSeconds() pero segundos UTC.
`now()`	Número de milisegundos desde el 1 de enero de 1970. (Date.now())
`parse()`	Parsea una cadena de fecha y devuelve el número de milisegundos desde el 1 de enero de 1970. (Date.parse("Aug 9, 1995"))
`toDateString()`	Convierte la fecha de un objeto Date en un string.
`toISOString()`	Covierte la fecha a string utilizando el estándar ISO.
`toJSON()`	Covierte la fecha a string a fecha con formato JSON.
`toLocaleDateString()`	Convierte la fecha de un objeto Date en un string usando la convención local.
`toLocaleTimeString()`	Convierte la hora de un objeto Date en un string usando la convención local.
`toLocaleString()`	Convierte un objeto Date en un string usando la convención local.
`toString()`	Convierte un objeto Date en un string.
`toTimeString()`	Convierte la hora de un objeto Date en un string.
`toUTCString()`	Convierte un objeto Date en un string de acuerdo al tiempo universal.
`UTC()`	Devuelve el número de milisegundos desde la medianoche del 1 de enero de 1970 de acuerdo a la hora UTC. (Date.UTC(año,mes[, dia[, hora[, minutos[, segundos, milisegundos]]]])
`valueOf()`	Retorna el valor primitivo de un objeto Date.

Expresiones regulares y el Objeto RegExp

- Las expresiones regulares son patrones de búsqueda que se pueden utilizar para encontrar texto que coincida con dicho patrón.
- En JavaScript las expresiones regulares se gestionan a través del objeto RegExp.
- Para crear un literal del tipo RegExp tendrás que usar la siguiente sintaxis:
  - var expresion = /expresión_regular/[flags];
    - var expresion=/Aloe\s+Vera/;

Características especiales y Flags en Expresiones regulares

Caracteres especiales utilizados en expresiones regulares			
Carácter	**Coincidencias**	**Patrón**	**Ejemplo de cadena**
`.`	Cualquier carácter excepto nueva línea	`/a.e/`	Que aparezca cualquier carácter, excepto nueva línea entre la a y la e: "ape" y "axe"
`\W`	Coincide con caracteres que **NO** sean (letras, dígitos, subrayados). El espacio en blanco no es considerado letra.	`/\W/`	Que aparezca un carácter (que no sea letra, dígito o subrayado): "%" en "100%"
`\w`	Coincide con caracteres del tipo (letras, dígitos, subrayados)	`/\w/`	Que aparezca un carácter (letra, dígito o subrayado): "J" en "JavaScript". No en "?!%"
`\D`	Cualquier carácter que **NO** sea un dígito	`/\D{2,4}/`	Que aparezcan mínimo 2 y máximo 4 caracteres que no sean dígitos: encontrará la cadena "Ahor" en "Tienes Ahora 45 años"
`\d`	Dígitos del 0 al 9	`/\d{3}/`	Que aparezcan exactamente 3 dígitos: encontrará la cadena "456" en "Ahora en 456"
`\B`	Coincide al final de una palabra	`/\Bno/`	Que "no" esté al final de una palabra: "este invierno" ("no" de "invierno")
	No coincide con el comienzo ni con el final de la palabra	`/\Bno\B/`	Palabras que contenga "no": renovado
`\b`	Coincide con el inicio de una palabra	`/\bno/`	Que "no" esté al comienzo de una palabra: "novedad"
	Coincide con el comienzo y el final de la palabra	`/\bno\b/`	La palabra "no" en "puede que no vaya". No coincidiría en "No hay novedades"
`[^…]`	Cualquier carácter excepto los que están entre corchetes	`/a[^px]e/`	Que aparezca cualquier carácter excepto la "p" o la "x" después de la letra a y antes de la letra e: "ale", pero no "axe" o "ape"
`[…]`	Cualquier carácter entre corchetes	`/a[px]e/`	Que aparezca alguno de los caracteres "p" o "x" entre la a y la e: "ape", "axe", pero no "ale"
`^`	Al inicio de una cadena	`/^Esto/`	Coincidencia en "Esto es..."
`$`	Al final de la cadena	`/final$/`	Coincidencia en "Esto es el final"
`*`	Coincide 0 o más veces	`/se*/`	Que la "e" aparezca 0 o más veces después de una letra s: "seeee" y también "se"
`?`	Coincide 0 o 1 vez	`/ap?/`	Que la p aparezca 0 o 1 vez después de la letra a: "apple" y

Características especiales y Flags en Expresiones regulares

Caracteres especiales utilizados en expresiones regulares			
Carácter	**Coincidencias**	**Patrón**	**Ejemplo de cadena**
`{n,m}`	Coincide al menos n, y máximo m veces	`/ap{2,4}/`	Que la "p" aparezca al menos 2 veces y como máximo 4 veces después de la letra a: "a**pppp**pple" (encontrará 4 "p")
`p1|p2`	Coincide con p1 (palabra 1) o p2 (palabra 2)	`/tu|mi/`	Contiene la palabra tu o mi: Pedro es mi nombre
`(…)`	Agrupa caracteres	`/(va)$/`	Termina en la agrupación de letras “va”: renueva
`\n`	Coincide con una nueva línea		
`\s`	Coincide con un espacio en blanco		
`\S`	Coincide con un carácter que **NO** es un espacio en blanco		
`\t`	Un tabulador		
`\r`	Un retorno de carro		

Flags o indicadores de las expresiones regulares

Flag	Significado
`g`	Coincidencia global: comprueba en toda la cadena, en lugar de detenerse cuando encuentra la primera coincidencia.
`i`	No es sensible a mayúsculas y/o minúsculas.
`m`	Se aplica el carácter especial de comenzar y terminar la línea (^ y $, respectivamente) a cada línea en una cadena de varias líneas. Para que lo haga en más de una línea hay que indicar también el flag g.

Objeto RegExp

- El objeto **RegExp** es tanto un literal como un objeto de JavaScript, por lo que también se podrá crear usando un constructor:
  - var expresionregular = new RegExp("Texto Expresión Regular");
- ¿Cuándo usar el literal o el objeto?
  - La expresión **RegExp** literal es compilada cuando se ejecuta el script, por lo tanto, se recomienda usar el literal cuando sabemos que la expresión no cambiará. Una versión compilada es mucho más eficiente.
  - Usaremos el objeto, cuando sabemos que la expresión regular va a cambiar, o cuando vamos a proporcionarla en tiempo de ejecución.

Objeto RegExp (Propiedades y Métodos)

Propiedades del objeto RegExp	
Propiedad	**Descripción**
`global`	Especifica que se utilizará el modificador "g".
`ignoreCase`	Especifica que se utilizará el modificador "i".
`lastIndex`	Índice donde comenzará la siguiente búsqueda.
`multiline`	Especifica si el modificador "m" es utilizado.
`source`	El texto de la expresión regular RegExp.

Métodos del objeto RegExp	
Método	**Descripción**
`compile()`	Compila una expresión regular.
`exec()`	Busca la coincidencia en una cadena. Devolverá la primera coincidencia.
`test()`	Busca la coincidencia en una cadena. Devolverá true o false.

Objeto RegExp (Ejemplos)

- Para comprobar si una cadena está incluida (contenida) en otra.

```
var datos = new Array();
datos[0] = "El Blogger de Google"; // aquí dará verdadero
datos[1] = "El blogger de Google"; // aquí dará falso (por la mayúscula B)
datos[2] = "BloggerGoogle";        // aquí dará verdadero
datos[3] = "Google Blogger";       // aquí dará falso (por el orden en el que están)
var patron = /Blog.*Goog/;  // Patrón de búsqueda que contenga en cualquier posición:
                            // Blog -la cadena "Blog"
                            // . -seguido de cualquier carácter excepto la nueva línea
                            // * -el carácter anterior 0 o más veces
                            // Goog -seguido de la cadena "Goog"
for (var i = 0; i < datos.length; i++)
  alert(datos[i] + " " + patron.test(datos[i]));
```

Objeto RegExp (Ejemplos)

- Validación de un número de Seguridad Social Americano
  - Un número de Seguridad Social americano consiste en 8 dígitos agrupados en tres campos separados, generalmente, por guiones: AAA-GG-SSS.
    - Los tres primeros dígitos (AAA), corresponden al "Número de área".
    - Los dos siguientes (GG), corresponden al "Número de grupo“.
    - Los 3 últimos (SSS), corresponden al "Número de serie“.
  - En este ejemplo se valida que un número facilitado cumpla el formato indicado con o sin guiones ya que éstos son opcionales (caracteres "-?" dentro de la expresión regular).

Objeto RegExp (Ejemplos)

```
function esCorrectoNSSAmericano(numero){
   var patron = /^\d{3}-?\d{2}-?\d{3}$/;
   patron.test(numero)?return true;:return false;
}
```


```
var unNumero=prompt(Introduce un n° de la seguridad social con formato americano\n
AAA-GG-SSS);
if (esCorrectoNSSAmericano(unNumero){
  alert("Correcto: el número "+numero+" cumple el estándar americano");
} else {
  alert("Error: el número "+numero+" NO cumple el estandar.");
}
```