Objetos Intrínsecos

Antonio Espín Herranz

Objetos intrínsecos

- **Intrísecos**: objetos predefinidos dentro de JavaScript, resultan de gran utilidad.
  - Se crean con **new**

- Dentro de estos objetos tenemos:
  - String, Number, Math y Date
  - Array
  - RegExp (*siguiente tema*)
  - Error
  - Map, Set
  - Function, Global y Object
  - Image

String: Cadenas

- Cuando se asocia una variable de texto:
  - var cadena = “hola”; // Se crea un objeto String de forma automática.

- Algunos métodos:
  - charAt(n): Retorna el carácter que ocupa la posición n. Empiezan en 0.
  - indexOf(cadena, inicio): Localizar cadena dentro del String. Inicio indica a partir de donde queremos empezar a buscar.
  - lastIndex(cadena): Retorna el índice correspondiente a la última aparición de la cadena.
  - substring(inicio, final): Extrae la subcadena comprendida entre los dos índices.

Métodos

Method	Description
charAt()	Returns the character at the specified index
charCodeAt()	Returns the Unicode of the character at the specified index
concat()	Joins two or more strings, and returns a copy of the joined strings
fromCharCode()	Converts Unicode values to characters
indexOf()	Returns the position of the first found occurrence of a specified value in a string
lastIndexOf()	Returns the position of the last found occurrence of a specified value in a string
localeCompare()	Compares two strings in the current locale
match()	Searches for a match between a regular expression and a string, and returns the matches
replace()	Searches for a match between a substring (or regular expression) and a string, and replaces the matched substring with a new substring
search()	Searches for a match between a regular expression and a string, and returns the position of the match
slice()	Extracts a part of a string and returns a new string

Métodos II

split()	Splits a string into an array of substrings
substring()	Extracts the characters from a string, between two specified indices
toLocaleLowerCase()	Converts a string to lowercase letters, according to the host's locale
toLocaleUpperCase()	Converts a string to uppercase letters, according to the host's locale
toLowerCase()	Converts a string to lowercase letters
toString()	Returns the value of a String object
toUpperCase()	Converts a string to uppercase letters
trim()	Removes whitespace from both ends of a string
valueOf()	Returns the primitive value of a String object

Método substr vs substring

- cadena.substr(inicio, num_chars)
  - Posición de inicio y a partir de esta posición corta num_chars

- cadena.substring(inicio, fin)
  - Del índice inicial al índice final

Más métodos (HTML)

- Insertan etiquetas HTML:
  - toLowerCase(): Transforma a minúsculas.
  - toUpperCase(): A mayúsculas.
  - big(): <big>
  - bold(): <b>
  - italics(): <i>
  - small(): Inserta la marca <small>
  - sub() / sup() → <sub> <sup>

```
var  cadena =“texto”;
alert(cadena.toUpperCase());
```

Ejemplo

- Función JavaScript que ordena una cadena.

```
<html>
<head> <title> Generación de páginas </title>
<script language=javascript>

function cambiar(cadena){
var orden=new Array();

for (var i=0;i < cadena.length;++i)
    orden[i]=cadena.charAt(i);

orden.sort();
return(orden.toString());
}

alert(cambiar(prompt("Dame cadena","")));

</script>
</head>
<body>
</body>
</html>
```

Pasamos todas las letras a un array. Lo ordenamos y convertimos otra vez a String.

Ejemplos

- var str = "Visit Microsoft!";
  var res = str.replace("Microsoft","W3Schools");
- **Visit W3Schools**

- var str = "Hello world!";
  var res = str.substr(1,4);
- **ello**

- var str = "Hello world!";
  var res = str.substring(1,4);
- **ell**

- var str = "Hello world!";
  var res = str.slice(1,5);
- The result of *res* will be:
- **ello**

- var str = "How are you doing today?";
  var res = str.split(" ");
- **Un array de 5 elementos.**

- var str = "Visit W3Schools!";
  var n = str.search("W3Schools");
- **6**

- var str = "The rain in SPAIN stays mainly in the plain";
  var res = str.match(/ain/g);
- **ain,ain,ain**

Números

- Objeto **Math**:
  - Nos permite realizar operaciones matemáticas.

- Objeto **Number**:
  - Nos proporciona las siguientes propiedades.
  - MAX_VALUE:
  - MIN_VALUE:
  - NEGATIVE_INFINITE:
  - POSITIVE_INFINITE:
  - NaN: Valor no numérico.
  - Para obtener la longitud de un número:
    - let x = 1234;
    - console.log(x.toString().length);

Math

- Permite el uso de propiedades y métodos asociados a las constantes y funciones matemáticas.
- Math.PI
- Math.sqrt(25);

- Funciones trigonométricas, etc.

Resolución de ecuaciones de 2 grado

Ejemplo: Resolver ec. De 2º grado

```
<HTML>
<HEAD>
<TITLE>Math</TITLE>
<SCRIPT LANGUAGE ="JavaScript">
<!-- se oculta la información de los navegadores antiguos

 function calculaEcuacion() {
  var a, b, c, solucion;

  a=parseFloat(document.miFormulario.a.value);
  b=parseFloat(document.miFormulario.b.value);
  c=parseFloat(document.miFormulario.c.value);

   solucion=(-b + Math.sqrt(Math.pow(b,2)-(4*a*c)))/(2*a);
   document.miFormulario.x1.value=solucion;
   solucion=(-b - Math.sqrt(Math.pow(b,2)-(4*a*c)))/(2*a);
   document.miFormulario.x2.value=solucion;
 }

// final del comentario -->
</SCRIPT>
</HEAD>
```

```
<BODY>
<H1>Manejando el objeto Math...</H1>
<FORM NAME="miFormulario">
<BR><B>Introduce los coeficientes de la</B>
<BR><B>siguiente ecuación de 2º grado:
    </B><BR><BR>
<INPUT TYPE=TEXT NAME="a" SIZE=3
    MAXLENGTH=3> x² +
<INPUT TYPE=TEXT NAME="b" SIZE=3
    MAXLENGTH=3> x +
<INPUT TYPE=TEXT NAME="c" SIZE=3
    MAXLENGTH=3> = 0<BR>
<BR><B>Las soluciones son:</B><BR><BR>
 x = <INPUT TYPE=TEXT NAME="x1" SIZE=5
    MAXLENGTH=5> ó
 x = <INPUT TYPE=TEXT NAME="x2" SIZE=5
    MAXLENGTH=5>
<BR><BR>
<INPUT TYPE=BUTTON VALUE="Solucionar"
    NAME="miBoton" onClick="calculaEcuacion()">
</FORM>
</BODY>
</HTML>
```

Métodos Math

Método	Descripción
abs	Valor Absoluto
sin, cos, tan	Funciones trigonométricas estandar; argumentos en radianes
acos, asin, atan, atan2	Funciones trigonométricas inversas; retornan valores en radianes
exp, log	Logaritmo exponencial y natural, base $_e$
ceil	Retorna un entero mayor o igual al argumento
floor	Retorna un entero menor o igual al argumento
min, max	Retorna el mayor o menor(respectivamente) de dos argumentos
pow	Exponencial; primer argumento es base, el segundo exponente
random	Retorna un número aleatorio entre 0 y 1.
round	Redondea el argumento al entero más cercano
sqrt	Raíz cuadrada

Date

- Permite la manipulación de datos que representan fechas.

- El constructor del objeto está sobrecargado.
  - Se puede invocar sin argumentos, para crear un objeto que contenga la fecha y la hora actual.
  - Una cadena con el siguiente formato:
    - “Mes día, año hora:minuto:segundo”
  - Tres enteros que representan el año, mes y día.
  - 6 enteros que representan el año, el mes, día, hora, los minutos y los segundos correspondientes.

  - var d = new Date();
    var d = new Date(*milliseconds*);
    var d = new Date(*dateString*);
    var d = new
    Date(*year*, *month*, *day*, *hours*, *minutes*, *seconds*, *milliseconds*);

- Ejemplo:

  var objetoFecha = new Date(2007, 05, 01, 23, 33, 55);

Métodos de Date

- Métodos **set** para modificar los distintos campos: Day, Hours, Month, etc. → setHours(12);

- Métodos **get** para extraer información de los distintos campos: getTime(), getYear(), …

- **UTC**(): Devuelve el número de milisegundos tomando como referencia el 1/1/70 00:00:00

- **toLocaleString**(): Convierte una fecha al formato local.

Ejemplo

```
<html>
<head><title>Fecha y Hora</title>

<script language=javascript>

function presen()
{
temp1=setTimeout("presen()",1000);
horas=new Date();
window.document.fecha.hora.value=horas.getHours();
window.document.fecha.min.value=horas.getMinutes();
window.document.fecha.seg.value=horas.getSeconds();
window.document.fecha.dia.value=horas.getDate();
window.document.fecha.mes.value=horas.getMonth()+1;
window.document.fecha.anio.value=horas.getYear();
}

</script>
</head>
```

```
<body onload="presen()">
<form name="fecha">
<input type=text name="hora" size=2>
   <b><font color="red" size=5> :</b>
<input type=text name="min"
   size=2><b><font color="red" size=5> :</b>
<input type=text name="seg" size=2>
<BR> <BR>
<input type=text name="dia" size=2> <b><font
   color="red" size=5>/</b>
<input type=text name="mes"
   size=2><b><font color="red" size=5>/</b>
<input type=text name="anio" size=4>
</form>
</body>
</html>
```

Métodos Date

Method	Description
getDate()	Returns the day of the month (from 1-31)
getDay()	Returns the day of the week (from 0-6)
getFullYear()	Returns the year (four digits)
getHours()	Returns the hour (from 0-23)
getMilliseconds()	Returns the milliseconds (from 0-999)
getMinutes()	Returns the minutes (from 0-59)
getMonth()	Returns the month (from 0-11)
getSeconds()	Returns the seconds (from 0-59)
getTime()	Returns the number of milliseconds since midnight Jan 1, 1970
getTimezoneOffset()	Returns the time difference between UTC time and local time, in minutes
getUTCDate()	Returns the day of the month, according to universal time (from 1-31)
getUTCDay()	Returns the day of the week, according to universal time (from 0-6)
getUTCFullYear()	Returns the year, according to universal time (four digits)
getUTCHours()	Returns the hour, according to universal time (from 0-23)

Métodos Date II

getUTCMilliseconds()	Returns the milliseconds, according to universal time (from 0-999)
getUTCMinutes()	Returns the minutes, according to universal time (from 0-59)
getUTCMonth()	Returns the month, according to universal time (from 0-11)
getUTCSeconds()	Returns the seconds, according to universal time (from 0-59)
getYear()	Deprecated. Use the getFullYear() method instead
parse()	Parses a date string and returns the number of milliseconds since midnight of January 1, 1970
setDate()	Sets the day of the month of a date object
setFullYear()	Sets the year (four digits) of a date object
setHours()	Sets the hour of a date object
setMilliseconds()	Sets the milliseconds of a date object
setMinutes()	Set the minutes of a date object
setMonth()	Sets the month of a date object
setSeconds()	Sets the seconds of a date object
setTime()	Sets a date and time by adding or subtracting a specified number of milliseconds to/from midnight January 1, 1970
setUTCDate()	Sets the day of the month of a date object, according to universal time
setUTCFullYear()	Sets the year of a date object, according to universal time (four digits)
setUTCHours()	Sets the hour of a date object, according to universal time

Métodos Date III

setUTCMilliseconds()	Sets the milliseconds of a date object, according to universal time
setUTCMinutes()	Set the minutes of a date object, according to universal time
setUTCMonth()	Sets the month of a date object, according to universal time
setUTCSeconds()	Set the seconds of a date object, according to universal time
setYear()	Deprecated. Use the setFullYear() method instead
toDateString()	Converts the date portion of a Date object into a readable string
toGMTString()	Deprecated. Use the toUTCString() method instead
toISOString()	Returns the date as a string, using the ISO standard
toJSON()	Returns the date as a string, formatted as a JSON date
toLocaleDateString()	Returns the date portion of a Date object as a string, using locale conventions
toLocaleTimeString()	Returns the time portion of a Date object as a string, using locale conventions
toLocaleString()	Converts a Date object to a string, using locale conventions
toString()	Converts a Date object to a string
toTimeString()	Converts the time portion of a Date object to a string
toUTCString()	Converts a Date object to a string, according to universal time
UTC()	Returns the number of milliseconds in a date string since midnight of January 1, 1970, according to universal time
valueOf()	Returns the primitive value of a Date object

Método UTC

- Se puede aplicar a la clase Date.

- Se le pasan los datos de la fecha y retorna el número de milisegundos desde el 01-01-1970 hasta la fecha indicada.

- Por ejemplo:
- console.log(Date.UTC(2022,11,2));

Array

- Representa una lista de datos, sean del tipo que sean.
- Formas de crear un Array:
  - Nombre_Array = new Array(longitud);
  - Nombre_Array = new Array(elem1, elem2, elemN);
  - Nombre_Array=[];
  - Nombre_Array = [elem1, elem2, …, elemN];

- Acceso a los elementos:
  - Nombre_Array[indice]; // Empiezan en 0 y soportan la propiedad length.

Métodos del Array

- **join(elemento_enlace**): nos devuelve un String resultado de la **concatenación** de todos los elems del Array, intercala el elemento_enlace.
- **toString**(): **Devuelve un String** con todos los elementos separados por una ,. Equivale a un **join(“,”);**
- **pop**(): **Elimina el último** elemento del array y devuelve el elemento.
- **shift**(): Idem del anterior pero actúa en el primer elemento del array. **Elimina el primero**.
- **push**(el1, el2, …, elN): **Añade** todos los elementos pasados por argumento.

Métodos del Array II

- **unshift**(el1, el2, … elN): Igual que push pero **añade por el principio.**
- **reverse**(): **Da la vuelta** al array. El último elemento se convierte en el primero.
- **slice**(lim_inf, lim_sup): **Extrae una porción** del array original.
- **sort**([funcion_compacion]): Para ordenación de arrays se puede pasar una función por parámetro para establecer el criterio de ordenación.

Métodos del Array III

- **splice**(index, cuantos, [val1,val2, …]): Añade o borra elementos de un arrays.
  - El index indica a partir de que posición añade o borra elementos.
  - Cuantos: El número de elementos a borrar. Con 0 no borra.

- Ejemplo:
- var fruits = ["Banana", "Orange", "Apple", "Mango"]; fruits.splice(2,0,"Lemon","Kiwi");
- Banana,Orange,Lemon,Kiwi,Apple,Mango

forEach

- El bucle forEach se define una función que recibe los parámetros del array, puede ser value o value e index, etc..
-
```
const numbers = [45, 4, 9, 16, 25];
let txt = "";
numbers.forEach(myFunction);

function myFunction(value, index, array) {
  txt += value + "<br>";
}
```

forEach en ES6

```
const numbers = [45, 4, 9, 16, 25];
let txt = "";

numbers.forEach((value)=>
   txt += value + "<br>”
);
```

Ejemplo: sort

- **Cambio el criterio de ordenación:**

```
function descendente(elem1,elem2){

    if (elem1 > elem2)
      return -1;

    else if (elem1 == elem2)
      return 0

    else
      return 1;
}

// Al llamar a sort:
orden.sort(descendente)
```

Ejemplo de sort ES6

```
const nombres = ['Ana', 'Sandra', 'Luis', 'Raul', 'Jorge'];
// Pruebas de ordenacion:
console.log("Antes de sort:", nombres);
nombres.sort((v1, v2) => {
    if (v1 < v2) {
        return -1;
    } else if (v1 == v2) {
        return 0;
    } else {
        return 1;
    }
});
console.log("Despues de sort:", nombres);
```

Arrays de dos dimensiones (I)

- Ejemplo
- var temperaturas_medias_ciudad0 = new Array(3)
  temperaturas_medias_ciudad0[0] = 12
  temperaturas_medias_ciudad0[1] = 10
  temperaturas_medias_ciudad0[2] = 11

  var temperaturas_medias_ciudad1 = new Array (3)
  temperaturas_medias_ciudad1[0] = 5
  temperaturas_medias_ciudad1[1] = 0
  temperaturas_medias_ciudad1[2] = 2

  var temperaturas_medias_ciudad2 = new Array (3)
  temperaturas_medias_ciudad2[0] = 10
  temperaturas_medias_ciudad2[1] = 8
  temperaturas_medias_ciudad2[2] = 10

Arrays de dos dimensiones (II)

- var temperaturas_cuidades = new Array (3)
  temperaturas_cuidades[0] = temperaturas_medias_ciudad0
  temperaturas_cuidades[1] = temperaturas_medias_ciudad1
  temperaturas_cuidades[2] = temperaturas_medias_ciudad2

- document.write("<table width=200 border=1 cellpadding=1 cellspacing=1>");
  for (i=0;i<temperaturas_cuidades.length;i++){
  document.write("<tr>")
  document.write("<td>Ciudad " + i + "</td>")
  for (j=0;j<temperaturas_cuidades[i].length;j++){
  document.write("<td>" + temperaturas_cuidades[i][j] + "</td>")
  }
  document.write("</tr>")
  }
  document.write("</table>")

Arrays const

- Son interesantes cuando no se tiene que modificar el contenido.
- Por ejemplo, los días de la semana, nombre de los meses …
- const meses = [‘ene’,’feb’,…];

Map

- Dentro de las colecciones disponemos de Map.
  - Reciben el nombre de arrays asociativos
  - Son pares de clave / valor
  - Las claves pueden ser texto no tienen que ser números como ocurre en los arrays.
  - Tienen una serie de métodos y la propiedad size para obtener el número de pares que tenemos en el mapa.
  - https://www.w3schools.com/js/js_maps.asp

Map métodos

Method	Description
new Map()	Creates a new Map object
set()	Sets the value for a key in a Map
get()	Gets the value for a key in a Map
clear()	Removes all the elements from a Map
delete()	Removes a Map element specified by a key
has()	Returns true if a key exists in a Map
forEach()	Invokes a callback for each key/value pair in a Map
entries()	Returns an iterator object with the [key, value] pairs in a Map
keys()	Returns an iterator object with the keys in a Map
values()	Returns an iterator object of the values in a Map

Property	Description
size	Returns the number of Map elements

Set

- Indica una colección de valores únicos (no permite mantener repetidos).

- Como Map mantienen una serie de métodos y también disponen de la propiedad size.

- https://www.w3schools.com/js/js_sets.asp

Set métodos

Method	Description
new Set()	Creates a new Set
add()	Adds a new element to the Set
delete()	Removes an element from a Set
has()	Returns true if a value exists
clear()	Removes all elements from a Set
forEach()	Invokes a callback for each element
values()	Returns an Iterator with all the values in a Set
keys()	Same as values()
entries()	Returns an Iterator with the [value,value] pairs from a Set

Property	Description
size	Returns the number elements in a Set

Error

- Se crean instancias cuando queremos lanzar excepciones personalizadas.

- Va ligado a la palabra reservada throw para lanzar y a la vez se crea el error con new.
  - Lo vemos en el tema de Excepciones.

Function, Global y Object

- **Global**: No se utiliza directamente, lo crea automáticamente el sistema. Este objeto recopila una serie de funciones como son: parseInt(), isNaN(), etc.
  - Se comentan en el apartado de funciones.

- **Object**: Está contenido en el resto de objetos. Una de sus propiedades es prototype y un método toString() estos están disponibles en todos los objetos (hasta los desarrollados por el programador)

- Para **Function** normalmente se implementarán no se suelen instanciar con new como los otros objetos intrínsecos.

Image

- Nos permite crear imágenes:
  - var myImage = new **Image**(100, 200);
  - myImage.src = 'picture.jpg';
  - console.log(myImage);

- Resultado:
  - <img width="100" height="200" src="picture.jpg">