UD3: Modelo de objetos predefinidos en Javascript

Desarrollo Web en Entorno Cliente

Javier G. Pisano
(jgonzalezp22@adistancia.educantabria.es)

Funciones

- Una **función** es un conjunto de instrucciones que se agrupan para realizar una tarea concreta y se pueden reutilizar de manera sencilla.
- Facilitan mucho la organización, y por consiguiente el mantenimiento y depuración de los programas.

Funciones simples

- Primero declaramos la función y luego la utilizamos (llamada o invocación):
  - Pueden recibir/devolver parámetros, pero lo veremos más adelante

```
/* Definición */
function nombreFuncion(){
sentencias;
}

/* Llamada o invocación */
nombreFuncion();
```

Funciones simples: Ejemplo

```
function sumayMuestra(){
 var resultado = numero1 + numero2;
 alert("El resultado es "+ resultado);
}
```

Definición

```
var resultado;
var numero1=3;
var numero2=5;

sumayMuestra();

numero1=5;
numero2=6;
sumayMuestra();
```

Llamada o invocación

Objetos

- Un objeto encapsula un conjunto de datos relacionados entre sí de modo que los puedo tratar de manera conjunta.
- Habitualmente en un objeto distinguimos:
  - Estado:
    - Contenido de las variables que lo forman.
    - A dichas variables las llamamos **propiedades**
  - Comportamiento:
    - Acciones (funciones) que puedo realizar con él.
    - A las funciones asociadas a un objeto las llamamos **métodos**.

Acceso a un objeto

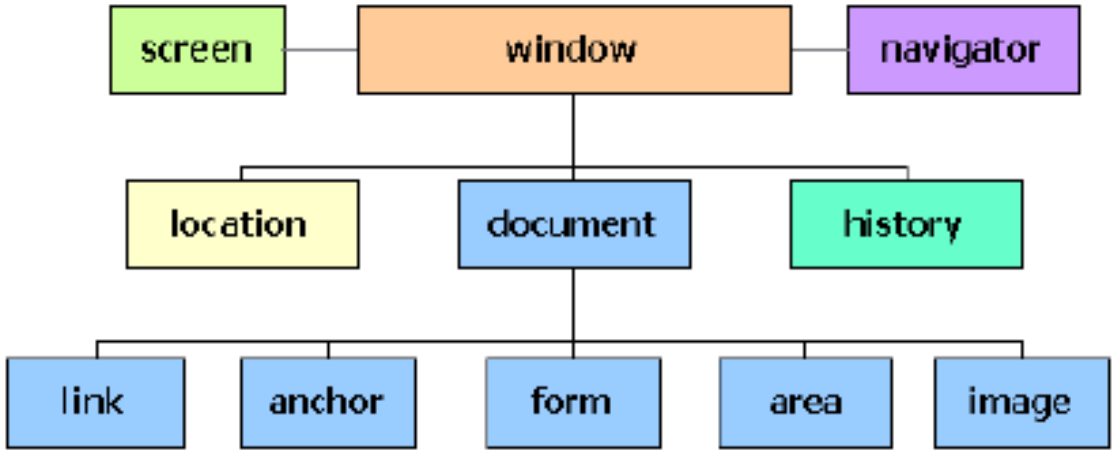
- Acceso a propiedades:
  - `nombreObjeto.propiedad`
- Acceso a un método de un objeto:
  - `nombreObjeto.método([parametros])`

DOM (*Document Object Model*)

- Habitualmente utilizamos Javascript para acceder al contenido de la página Web.
- Desde Javascript puedo acceder a una estructura de datos que representa (en forma de árbol) todo el contenido de una página Web.
- Es posible:
  - Leer datos de la estructura
  - Modificar la estructura (añado contenido a la página Web).
    - Modificar elementos.
    - Añadir elementos.

Objetos de alto nivel

- El gráfico simplificado del DOM es el siguiente:



Objeto `window`

- Es el objeto de más alto nivel y por tanto el contenedor principal de todo el contenido que se visualiza en el navegador.
- Representa cada ventana/pestaña
- Acceso:
  - `window.nombrePropiedad`
  - `windows.nombreMetodo([parámetros])`
- *Como* `window` contiene el resto de objetos, podemos omitir su nombre para acceder a propiedades y métodos dentro de esa ventana:
  - `nombrePropiedad`
  - `nombreMetodo([parámetros])`

Propiedades de `window`

Propiedad	Descripción
closed	Devuelve un valor Boolean indicando cuando una ventana ha sido cerrada o no.
defaultStatus	Ajusta o devuelve el valor por defecto de la barra de estado de una ventana.
document	Devuelve el objeto document para la ventana.
frames	Devuelve un array de todos los marcos (incluidos iframes) de la ventana actual.
history	Devuelve el objeto history de la ventana.
length	Devuelve el número de frames (incluyendo iframes) que hay en dentro de una ventana.
location	Devuelve la Localización del objeto ventana (URL del fichero).
name	Ajusta o devuelve el nombre de una ventana.
navigator	Devuelve el objeto navigator de una ventana.
opener	Devuelve la referencia a la ventana que abrió la ventana actual.
parent	Devuelve la ventana padre de la ventana actual.
self	Devuelve la ventana actual.
status	Ajusta el texto de la barra de estado de una ventana.

Métodos de **window**

Método	Descripción
alert()	Muestra una ventana emergente de alerta y un botón de aceptar.
blur()	Elimina el foco de la ventana actual.
clearInterval()	Resetea el cronómetro ajustado con setInterval().
setInterval()	Llama a una función o evalúa una expresión en un intervalo especificado (en milisegundos).
close()	Cierra la ventana actual.
confirm()	Muestra una ventana emergente con un mensaje, un botón de aceptar y un botón de cancelar.
focus()	Coloca el foco en la ventana actual.
open()	Abre una nueva ventana de navegación.
prompt()	Muestra una ventana de diálogo para introducir datos.

Intervalos de tiempo (1)

- Función **`setTimeout(nombreFuncion,miliseg)`**
  - Permite ejecutar una función una vez que haya transcurrido un periodo de tiempo determinado.
  - Si quiero que se ejecute varias veces debo invocarla de nuevo
  - OJO: El nombre de la función va sin paréntesis

```
setTimeout(refresca, 1000);

function refresca(){
hazAlgo();
/* Si quiero que se ejecute de nuevo: */
setTimeout(refresca,1000);
}

function hazAlgo(){
/* Hago algo */
}
```

Intervalos de tiempo (2)

- Función **`setInterval(nombreFuncion,miliseg)`**
  - Permite ejecutar una función cada intervalo de tiempo especificado.
  - NO es necesario especificar de nuevo la función para que se repita:

```
setInterval(hazAlgo, 1000);

function hazAlgo(){
/* Hago algo */
}
```

  - Podemos resetear el cronómetro usando `clearInterval()`

Objeto `location`

- Contiene información referente a la URL actual
- Podemos acceder con **`window.location`** o **`location`**

Propiedades de `location`

Propiedad	Descripción
hash	Cadena que contiene el nombre del enlace, dentro de la URL.
host	Cadena que contiene el nombre del servidor y el número del puerto, dentro de la URL.
hostname	Cadena que contiene el nombre de dominio del servidor (o la dirección IP), dentro de la URL.
href	Cadena que contiene la URL completa.
pathname	Cadena que contiene el camino al recurso, dentro de la URL.
port	Cadena que contiene el número de puerto del servidor, dentro de la URL.
protocol	Cadena que contiene el protocolo utilizado (incluyendo los dos puntos), dentro de la URL.
search	Cadena que contiene la información pasada en una llamada a un script, dentro de la URL

Métodos de `location`

Método	Descripción
assign()	Carga un nuevo documento.
reload()	Vuelve a cargar la URL especificada en la propiedad href del objeto location
replace()	Reemplaza el historial actual mientras carga la URL especificada en cadenaURL.

Objeto `navigator`

- Contiene información sobre el navegador que estamos usando cuando abrimos una URL o documento local

Propiedades de `navigator`

Propiedad	Descripción
appCodeName	Cadena que contiene el nombre en código del navegador.
appName	Cadena que contiene el nombre del cliente.
appVersion	Cadena que contiene información sobre la versión del cliente.
cookieEnabled	Determina si las cookies están o no habilitadas en el navegador.
platform	Cadena con la plataforma sobre la que se está ejecutando el programa cliente.
userAgent	Cadena que contiene la cabecera completa del agente enviada en una petición HTTP. Contiene la información de las propiedades appCodeName y appVersion.

Métodos de **navigator**

Método	Descripción
javaEnabled()	Devuelve true si el cliente permite la utilización de Java, en caso contrario, devuelve false.

Objeto **`document`**

- Representa el documento cargado en una ventana del navegador
- Representa el **acceso a todas las etiquetas HTML** dentro de una página
- Forma parte del objeto **`window`**, luego puede ser accedido mediante
  - `window.document`
  - `document`

Colecciones de **document**

Colección	Descripción
anchors[]	Es un array que contiene todos los hiperenlaces del documento.
forms[]	Es un array que contiene todos los formularios del documento.
images[]	Es un array que contiene todas las imágenes del documento.
links[]	Es un array que contiene todos los enlaces del documento.

Propiedades de `document`

Propiedad	Descripción
cookie	Devuelve todos los nombres/valores de las cookies en el documento.
domain	Cadena que contiene el nombre de dominio del servidor que cargó el documento.
referrer	Cadena que contiene la URL del documento desde el cuál llegamos al documento actual.
title	Devuelve o ajusta el título del documento.
URL	Devuelve la URL completa del documento.

Métodos de **document**

Método	Descripción
close()	Cierra el flujo abierto previamente con document.open().
getElementById()	Para acceder a un elemento identificado por el id escrito entre paréntesis.
getElementsByName()	Para acceder a los elementos identificados por el atributo name escrito entre paréntesis.
getElementsByTagName()	Para acceder a los elementos identificados por el tag o la etiqueta escrita entre paréntesis.
open()	Abre el flujo de escritura para poder utilizar document.write() o document.writeln en el documento.
write()	Para poder escribir expresiones HTML o código de JavaScript dentro de un documento.
writeln()	Lo mismo que write() pero añade un salto de línea al final de cada instrucción.

Objeto `frame`

- Representa un marco HTML
  - No se recomienda el uso de marcos, acarrea problemas de usabilidad y accesibilidad.
- Identifica una ventana particular dentro de un conjunto de marcos
- Para cada etiqueta `<frame>` se creará un objeto frame
  - Los iframe se comportan igual que los frames

Propiedades de `frame`

Propiedad	Descripción
align	Cadena que contiene el valor del atributo align (alineación) en un iframe.
contentDocument	Devuelve el objeto documento contenido en un frame/iframe.
contentWindow	Devuelve el objeto window generado por un frame/iframe.
frameBorder	Cadena que contiene el valor del atributo frameborder (borde del marco)
height	Cadena que contiene el valor del atributo height (altura)
longDesc	Cadena que contiene el valor del atributo longdesc (descripción larga)
marginHeight	Cadena que contiene el valor del atributo marginheight (alto del margen)
marginWidth	Cadena que contiene el valor del atributo marginwidth (ancho del margen)
name	Cadena que contiene el valor del atributo name (nombre)
noResize	Cadena que contiene el valor del atributo noresize
scrolling	Cadena que contiene el valor del atributo scrolling (desplazamiento)
src	Cadena que contiene el valor del atributo src (origen)
width	Cadena que contiene el valor del atributo width (ancho)

Propiedades de `frame`

Propiedad	Descripción
align	Cadena que contiene el valor del atributo align (alineación) en un iframe.
contentDocument	Devuelve el objeto documento contenido en un frame/iframe.
contentWindow	Devuelve el objeto window generado por un frame/iframe.
frameBorder	Cadena que contiene el valor del atributo frameborder (borde del marco)
height	Cadena que contiene el valor del atributo height (altura)
longDesc	Cadena que contiene el valor del atributo longdesc (descripción larga)
marginHeight	Cadena que contiene el valor del atributo marginheight (alto del margen)
marginWidth	Cadena que contiene el valor del atributo marginwidth (ancho del margen)
name	Cadena que contiene el valor del atributo name (nombre)
noResize	Cadena que contiene el valor del atributo noresize
scrolling	Cadena que contiene el valor del atributo scrolling (desplazamiento)
src	Cadena que contiene el valor del atributo src (origen)
width	Cadena que contiene el valor del atributo width (ancho)

Comunicación entre marcos

- El documento padre contiene un array con sus marcos hijo
  - Podemos acceder a un marco a través de:
    - Sintaxis de array
      - `[window].frames[n].title`
    - Nombre/id del marco
      - `[window].frames[“frameidzo”].title`
    - Atributo name de `<frame>`
      - `frameizdo.title`

Objetos nativos en Javascript

- Están listos para su utilización en nuestra aplicación
- Principales objetos:
  - String
  - Math
  - Number
  - Boolean

String

- Uno o más caracteres de texto rodeados de comillas simples o dobles.
- Cuando tenemos cadenas largas podemos concatenarlas con +=

```
var nuevoDocumento = "";
nuevoDocumento += "<!DOCTYPE html>";
nuevoDocumento += "<html>" ;
nuevoDocumento += "<head>";
nuevoDocumento += '<meta http-equiv="content-type"';
nuevoDocumento += ' content="text/html;charset=utf-8">';
```

Creación de `String`

- Sintaxis
  - `var miCadena="texto de la cadena";`
- Creación alternativa (sintaxis de objeto)
  - `var miCadena=new String("texto de la cadena");`

Propiedades de `String`

Propiedad	Descripción
length	Devuelve la longitud de una cadena.

Métodos de `String`

Métodos	Descripción
charAt()	Devuelve el carácter especificado por la posición que se indica por parámetro.
charCodeAt()	Devuelve el Unicode del carácter especificado por la posición que se
concat()	Une una o más cadenas y devuelve el resultado de esa unión.
fromCharCode()	Convierte valores Unicode a caracteres.
indexOf()	Devuelve la posición de la primera ocurrencia del carácter buscado en la cadena.
lastIndexOf()	Devuelve la posición de la última ocurrencia del carácter buscado en la cadena.
match()	Busca una coincidencia entre una expresión regular y una cadena y devuelve las coincidencias o null si no ha encontrado nada.
replace()	Busca una subcadena en la cadena y la reemplaza por la nueva cadena especificada.
search()	Busca una subcadena en la cadena y devuelve la posición dónde se encontró.
slice()	Extrae una parte de la cadena y devuelve una nueva cadena.
split()	Divide una cadena en un array de subcadenas.
substr()	Extrae los caracteres de una cadena, comenzando en una determinada posición y con el número de caracteres indicado.
substring()	Extrae los caracteres de una cadena entre dos índices especificados.
toLowerCase()	Convierte una cadena en minúsculas.
toUpperCase()	Convierte una cadena en mayúsculas.

Métodos de `String`

Métodos	Descripción
charAt()	Devuelve el carácter especificado por la posición que se indica por parámetro.
charCodeAt()	Devuelve el Unicode del carácter especificado por la posición que se
concat()	Une una o más cadenas y devuelve el resultado de esa unión.
fromCharCode()	Convierte valores Unicode a caracteres.
indexOf()	Devuelve la posición de la primera ocurrencia del carácter buscado en la cadena.
lastIndexOf()	Devuelve la posición de la última ocurrencia del carácter buscado en la cadena.
match()	Busca una coincidencia entre una expresión regular y una cadena y devuelve las coincidencias o null si no ha encontrado nada.
replace()	Busca una subcadena en la cadena y la reemplaza por la nueva cadena especificada.
search()	Busca una subcadena en la cadena y devuelve la posición dónde se encontró.
slice()	Extrae una parte de la cadena y devuelve una nueva cadena.
split()	Divide una cadena en un array de subcadenas.
substr()	Extrae los caracteres de una cadena, comenzando en una determinada posición y con el número de caracteres indicado.
substring()	Extrae los caracteres de una cadena entre dos índices especificados.
toLowerCase()	Convierte una cadena en minúsculas.
toUpperCase()	Convierte una cadena en mayúsculas.

Objeto `Math`

- Permite realizar operaciones matemáticas
- No posee constructor
  - No creamos instancias de objetos de tipo Math
  - Sus métodos y propiedades son **estáticas** (pertenecen a la clase, no al objeto).

```
var x = Math.PI; // Devuelve el número PI.
var y = Math.sqrt(16); // Calcula la raíz cuadrada de 16.
```

Propiedades de `Math`

Propiedad	Descripción
E	Devuelve el número Euler (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 base 2 de E (aproximadamente 1.442).
LOG10E	Devuelve el logaritmo base 10 de E (aproximadamente 0.434).
PI	Devuelve el número PI (aproximadamente 3.14159).
SQRT2	Devuelve la raíz cuadrada de 2 (aproximadamente 1.414).

Métodos de `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 alta hacia el siguiente entero.
cos(x)	Devuelve el coseno de x (x está en radianes).
floor(x)	Devuelve el número x redondeado a la baja hacia el anterior entero.
log(x)	Devuelve el logaritmo neperiando (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.

Objeto **Number**

- Es un envoltorio numérico para tipos numéricos primitivos

Propiedades de **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).
POSITIVE_INFINITY	Representa a infinito positivo (se devuelve en caso de overflow).
prototype	Permite añadir nuestras propias propiedades y métodos a un objeto.

Métodos de **Number**

Método	Descripción
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 la longitud x.
toString()	Convierte un objeto Number en una cadena. • Si se pone 2 como parámetro se mostrará el número en binario. • Si se pone 8 como parámetro se mostrará el número en octal. • Si se pone 16 como parámetro se mostrará el número en hexadecimal.
valueOf()	Devuelve el valor primitivo de un objeto Number.

Objeto **Boolean**

- Convierte a valores booleanos
- Propiedades

Propiedad	Descripción
constructor	Devuelve la función que creó el objeto Boolean.
prototype	Te permitirá añadir propiedades y métodos a un objeto

- Métodos

Método	Descripción
toString()	Convierte un valor Boolean a una cadena y devuelve el resultado.
valueOf()	Devuelve el valor primitivo de un objeto Boolean.

Objeto **Date**

- Se usa para trabajar con fechas y horas
- Constructores:

```
var d = new Date(); // Crea objeto fecha con la actual
var d = new Date(cadena de Fecha);
// Crea objeto con la fecha de cadena

var d = new Date(año, mes, día, horas, minutos, segundos,
milisegundos);
// Crea objeto fecha y hora con los parámetros
// el mes comienza en 0, enero será el 0
```

Propiedades de `Date`

Propiedad	Descripción
constructor	Devuelve la función que creó el objeto Date.
prototype	Te permitirá añadir propiedades y métodos a un objeto.

Métodos de **Date**

Métodos	Propiedades
getDate()	Devuelve el día del mes (de 1-31).
getDay()	Devuelve el día de la semana (de 0-6).
getFullYear()	Devuelve el año (4 dígitos).
getHours()	Devuelve la hora (de 0-23).
getMilliseconds()	Devuelve los milisegundos (de 0-999).
getMinutes()	Devuelve los minutos (de 0-59).
getMonth()	Devuelve el mes (de 0-11).
getSeconds()	Devuelve los segundos (de 0-59).
getTime()	Devuelve 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()	Devuelve 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()	Devuelve el año en base a la hora UTC (4 dígitos).
setDate()	Ajusta el día del mes del objeto (de 1-31).
setFullYear()	Ajusta el año del objeto (4 dígitos).
setHours()	Ajusta la hora del objeto (de 0-23).

Clase **Array**

- Cuando definimos un array en Java realmente estamos definiendo un objeto de la clase **`Array`**.
- Por tanto, las siguientes sentencias son completamente equivalentes:

```
var mascotas=[];
```

```
var mascotas=new Array();
```

- Podemos acceder al tamaño de un array con la propiedad `length`.

```
alert(mascotas.length);
```

Clase `Array`

- La clase también define (entre otros) los métodos:
  - `push()`: Añade un elemento al final del array.
  - `pop()`: Retira un elemento del final del array.

```
var mascotas=[];

mascotas.push("pajaro");
mascotas.push("tortuga");

alert(mascotas[0]);
alert(mascotas.length);
```

Objeto `console`

- Ofrece una interfaz para comunicarnos por código con la consola
  - No es estándar pero viene incluida en la mayor parte de los navegadores.

```
/* AVISOS (WARNING) */
console.warn("Mensaje");

/* ERRORES */
console.error("Mensaje");

/* INFORMACIÓN (LOG) */
console.log("Mensaje");

/* DEPURACIÓN (DEBUG) */
console.debug("Mensaje");
```

Objeto **console** : Ejemplo

```
var numero=prompt("Introduzca un número positivo");

if(numero>0){
 console.debug("Es positivo");
 hazCosas();
}
else{
 console.debug("Es negativo");
 hazOtrasCosas();
}

console.debug("Fin del programa");
```