UD3: Objetos predefinidos en Javascript

Desarrollo Web en Entorno Cliente

Índice

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**.

Clase: Agrupa un conjunto de objetos con estado y comportamiento común

Acceso a un objeto

- Creación de un objeto (una instancia)
  - `var nombreObjeto=new NombreClase()`
- Acceso a propiedades:
  - `nombreObjeto.propiedad`
- Acceso a un método de un objeto:
  - `nombreObjeto.método([parametros])`

Objetos nativos

Indice

UD3: Objetos predefinidos en JS

Creación de `String`

- Sintaxis tradicional

```
var miCadena="texto de la cadena";
```

- Creación alternativa (sintaxis de objeto)

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

Independientemente de la sintaxis usada podemos acceder a los métodos y propiedades

Propiedades y métodos de `String`

- Propiedades

`length`	Longitud de la cadena

- Métodos

`charAt(pos)`	Devuelve el carácter ubicado en **pos**
`charCodeAt(pos)`	Devuelve el Unicode del caracter ubicado en **pos**
`fromCharCode(code)`	Convierte valores Unicode a caracteres

Métodos de `String`

`concat(c1,c2)`	Concatena **`c1`** y **`c2`** devolviendo el resultado.
`indexOf(car)`	Devuelve la posición de la primera ocurrencia del carácter buscado por **`car`**
`lastIndexOf(car)`	Devuelve la posición de la última ocurrencia del carácter buscado por **`car`**
`match(er)`	Busca una coincidencia entre una expresión regular y la cadena, devolviendo las coincidencias
`replace(c1,c2)`	Busca la subcadena **`c1`** y la reemplaza con **`c2`**
`search(c)`	Busca la subcadena **`c`** y devuelve la posición donde se encontró

Métodos de `String`

`slice(inicio,fin)`	Extrae y devuelve la subcadena entre los índices **`inicio`** y **`fin`**.
`split(separador)`	Devuelve un array de subcadenas. El parámetro especifica el carácter a usar para la separación de la cadena.
`substr(inicio,[lon])`	Devuelve la subcadena que comienza en **`inicio`**
`substring(i1,i2)`	Devuelve la subcadena entre **`i1`** e **`i2`**
`toLowerCase()`	Convierte la cadena a minúsculas
`toUpperCase()`	Convierte la cadena a mayúsculas

String (Ejemplos)

```
var cadena="El parapente es un deporte de riesgo";
console.log("La longitud de la cadena es "+cadena.length);
console.log(cadena.toLowerCase());
console.log(cadena.charAt(3));
console.log(cadena.indexOf("pente"));
console.log(cadena.substring(3,16));
```

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`

`E`	Número e (aprox. 2.78)
`LN2`	Logaritmo neperiano de 2 (aprox. 0.69)
`LN10`	Logaritmo neperiano de 10 (aprox. 2.3)
`LOG2E`	Logaritmo en base 2 de e (aprox. 1.44)
`LOG10E`	Logaritmo en base 10 de e (aprox. 0.43)
`PI`	Número PI (aprox. 3.14)
`SQRT2`	Raíz cuadrada de 2 (aprox. 1.41)

Métodos de Math

abs(x)	Valor absoluto de **x**
ceil(x)	Número **x** redondeado al alza al siguiente entero
floor(x)	Número **x** redondeado a la baja al anterior entero
round(x)	Redondea **x** al entero más próximo
random()	Devuelve un numero aleatorio entre 0 y 1
pow(x,y)	Devuelve el resultado de **x** elevado a **y**
log(x)	Logaritmo neperiano de **x**
sqrt(x)	Raíz cuadrada de **x**
max(x,y,z...n) **min(x,y,z...n)**	Máximo/mínimo de los números que se pasan como parámetros
sin(x) **cos(x)** **tan(x)**	Funciones trigonométricas

Objeto Number
Propiedades

- Es un envoltorio numérico para tipos numéricos primitivos
- Propiedades estáticas:

MAX_VALUE	Número más alto posible
MIN_VALUE	Número más bajo posible
NEGATIVE_INFINITY	Infinito negativo (en caso de overflow)
POSITIVE_INFINITY	Infinito positivo (en caso de overflow)

Métodos de Number

toExponential(n)	Devuelve una cadena que representa el número en notación exponencial usando **n** dígitos en la parte decimal
toFixed(n)	Devuelve el número usando **n** dígitos decimales. Si no se especifica **n** por defecto se devuelve el número entero.
toPrecision(n)	Devuelve el número usando **n** dígitos decimales. Si no se especifica n por defecto se devuelve el número entero.
toString(B)	Representación como cadena en base B

Clase Date

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

`Date()`	Crea un objeto **`Date`** con la fecha actual
`Date(cadena)`	Crea un objeto **`Date`** a partir de la información de cadena
`Date(a,m,d,h,m,s,ms)`	Crea un objeto **`Date`** a partir del año, mes, día, hora, minuto, segundo y milisegundo.

Métodos de Date

getDate()	Devuelve el día del mes (1-31)
getDay()	Devuelve el día de la semana (0-6)
getFullYear()	Devuelve el año (4 dígitos)
getHours()	Devuelve la hora (0-23)
getMiliseconds()	Devuelve los milisegundos (0-9999)
getMinutes()	Devuelve los segundos (0-59)
getMonth()	Devuelve el mes (0-11)
getSeconds()	Devuelve los segundos (0-59)

Métodos de Date

getTime()	Devuelve los milisegundos desde el 1 de enero de 1970 (timestamp)
getTimeZoneOffset()	Diferencia en minutos entre hora GTM y hora local
setDate()	Ajusta el día del mes (1-31)
setFullYear()	Ajusta el año (4 dígitos)
setHour()	Ajusta la hora (0-23)

Clase Array (Introducción)

- Cuando definimos un array en Javascript realmente estamos definiendo un objeto de la clase **Array**.
- Podemos inicializarlo de distintas maneras:

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

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

```
var mascotas=["Pepi","Luci","Bom"];
```

```
var mascotas=new Array("Pepi","Luci","Bom");
```

```
var mascotas=new Array(10);
/* Se inicializan los elementos a null*/
```

Propiedades y métodos de `Array` (Uso básico)

- Propiedades

`length`	Longitud del array

- Métodos

`pop()`	Elimina el último elemento del array y lo devuelve
`push(elemento)`	Añade un elemento al final del array y devuelve la nueva longitud

EJERCICIO PROPUESTO

- Crea un programa que pida al usuario su nombre y apellidos y muestre:
  - El tamaño del nombre más los apellidos (sin contar espacios).
  - La cadena en minúsculas y en mayúsculas.
  - Que divida el nombre y los apellidos y los muestre en 3 líneas, donde ponga Nombre: / Apellido 1: / Apellido 2:
  - Una propuesta de nombre de usuario, compuesto por la inicial del nombre, el primer apellido y la inicial del segundo apellido: ej. Para Antonio Sierra García sería antoniosg.

EJERCICIO PROPUESTO

- Crea un programa que muestre el número de días que quedan desde hoy hasta el fin de curso (por ejemplo, el 24 de junio).

EJERCICIO PROPUESTO

- Crea un programa que capitalice la primera letra de cada palabra de una cadena

BOM (*Browser Object Model*)

Indice

UD3: Objetos predefinidos en JS

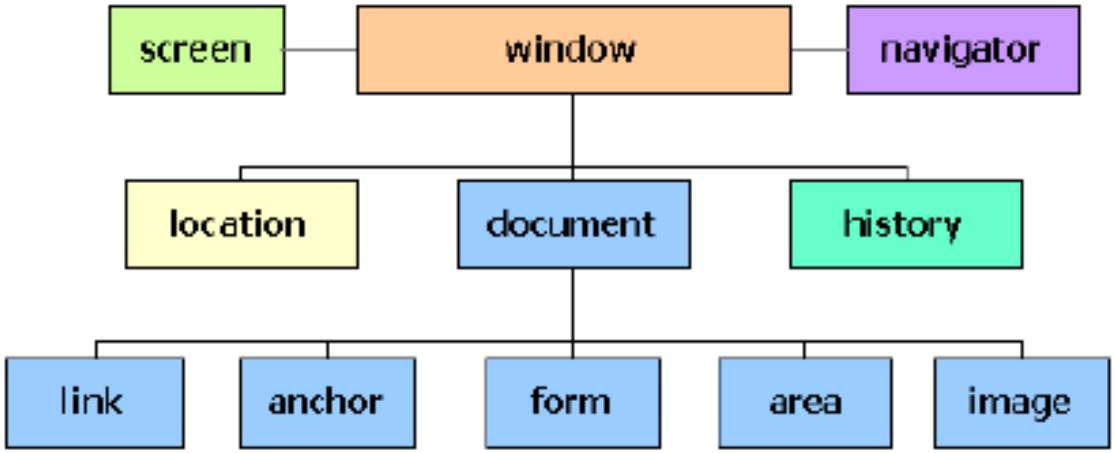
BOM (*Browser Object Model*)

- Permite acceder y modificar las propiedades de las ventanas del propio navegador:
  - Redimensionar y mover la ventana del navegador
  - Modificar el texto de la barra de estado
  - Acceder a la URL
  - Obtener información sobre el propio navegador…

PROBLEMA: Ninguna entidad se encarga de estandarizarlo, garantizando mínimos de compatibilidad entre navegadores

Objetos de alto nivel

- El BOM está formado por varios objetos relacionados entre sí
- Aquí se muestran los principales objetos:



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`
  - `window.nombreMetodo([parámetros])`

Objeto window

- Podemos omitir su nombre para acceder a propiedades y métodos dentro de esa ventana:
  - `nombrePropiedad`
  - `nombreMetodo([parámetros])`

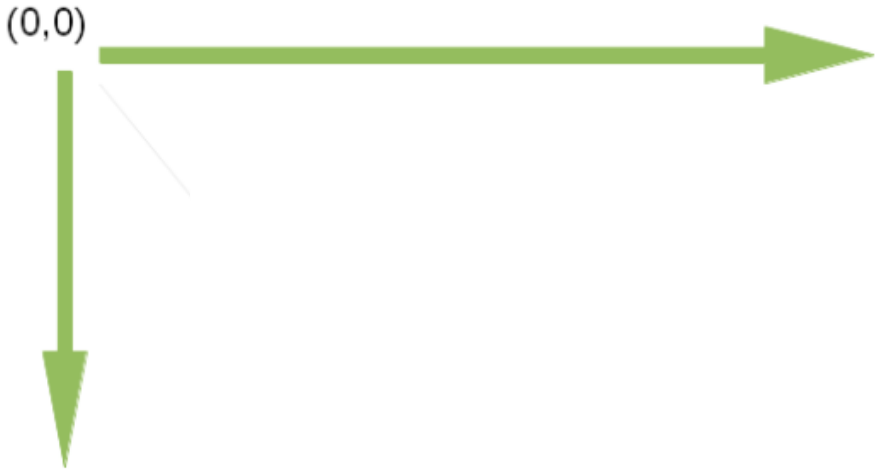
Manipulando el tamaño de la ventana

- Se definen 4 métodos en `window` para manipular tamaño y posición de la ventana:

`moveBy(x,y)`	Desplaza la posición de la ventana x píxeles a la derecha e y píxeles hacia abajo
`moveTo(x,y)`	Desplaza la ventana hasta que la esquina superior izquierda esté en (x,y)
`resizeBy(x,y)`	Redimensiona la ventana añadiendo x píxeles a la anchura anterior e y píxeles a la altura anterior
`resizeTo(x,y)`	Redimensiona la ventana para que su anchura sea igual a x y su altura sea igual a y.

Desplazamientos

- Generalmente medimos los desplazamientos en píxeles `(x,y)` desde la esquina superior izquierda, que representa el origen de coordenadas
  - **x** positivo: Derecha
  - **x** negativo: Izquierda
  - **y** positivo: Abajo
  - **y** negativo: Arriba



Manipulando el tamaño de la ventana

- Los navegadores cada vez son menos permisivos con la **modificación mediante JS** de las propiedades de sus ventanas
  - La mayoría de los navegadores permiten a los usuarios bloquear mediante JS este tipo de cambios.
  - Por ello, nunca debemos asumir que este tipo de funciones están disponibles.

Averiguando el tamaño de la ventana

- Cada navegador implementa su propio mecanismo:

Internet Explorer	**window.screenLeft** **window.screenTop**	Posición de la ventana.
	document.body.offsetWidth **document.body.offsetHeight**	Tamaño de la ventana (área visible).
Mozilla Safari Opera Chrome	**window.screenX** **window.screenY**	Posición de la ventana.
	window.innerWidth **window.innerHeight**	Tamaño de la ventana (área visible).
	window.outerHeight **window.outerWidth**	Tamaño de la ventana (total, con barra de estado y menús).

Intervalos de tiempo

- Función `setTimeout(nombreFuncion,miliseg)`
  - Permite ejecutar una función **una vez que haya transcurrido** un periodo de tiempo determinado.
- Función `setInterval(nombreFuncion,miliseg)`
  - Permite ejecutar una función **cada** intervalo de tiempo especificado.

OJO! La referencia a `nombreFuncion` se especifica sin paréntesis
De lo contrario estamos haciendo una llamada a la misma

Intervalos de tiempo

```
setTimeout(hazAlgo, 1000);

function hazAlgo(){
 console.log(“Hola");
}
```

```
setInterval(hazAlgo, 1000);

function hazAlgo(){
 console.log(“Hola");
}
```

SINTAXIS 1 Función nominal

SINTAXIS2 Función anónima

```
setTimeout(function(){ console.log("Muestra mensaje");}, 1000);
```

Parando el timeout

- La llamada a `setTimeout()/setInterval()` devuelve un identificador de reloj
- Si queremos impedir que se ejecute, podemos usar dicho identificador junto con la función `clearTimeout()/clearInterval()`

Parando el timeout

```
function muestraMensaje() {
  alert("Han transcurrido 3 segundos");
}
var id = setTimeout(muestraMensaje, 3000);


// Cancelamos la ejecución antes de que pasen 3seg
clearTimeout(id);
```

Parando el intervalo

```
function muestraMensaje() {
  alert("Aparezco cada segundo");
}
var id = setInterval(muestraMensaje, 1000);


// Cancelamos la ejecución del intervalo
clearInterval(id);
```

Intervalos de tiempo y parámetros

- Si queremos pasar parámetros a las funciones de intervalo podemos hacerlo añadiendo la lista de parámetros tras el tiempo
- **setTimeout(nombreFuncion,miliseg,[param1,param2…])**
- **setInterval(nombreFuncion,miliseg,[param1,param2…])**

```
function muestraMensaje(mensaje) {
  alert(mensaje);
}

setTimeout(muestraMensaje, 1000, "Hola");
```

Otros métodos de `window`

`close()`	Cierra la ventana actual
`open(url)`	Abre una nueva ventana que carga la URL especificada
`focus()`	Coloca el foco en la ventana actual
`blur()`	Elimina el foco de la ventana actual
`alert()`	Muestra un mensaje con un botón de aceptar
`prompt()`	Muestra una ventana de diálogo para introducir datos
`confirm()`	Muestra un mensaje con un botón de aceptar y otro de cancelar

Objeto `location`

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

Propiedades de `location`

`href`	URL completa
`host`	Nombre del servidor
`pathname`	Contenido tras el host
`hash`	Contenido tras # (secciones)
`port`	Puerto
`protocol`	Protocolo
`query`	Contenido tras ? (*query string*)

`href` es la más usada para leer o establecer la dirección de la página que muestra el navegador

Ejemplo

href	http://www.ejemplo.com/ruta1/ruta2/pagina.html#seccion
host	www.ejemplo.com
pathname	ruta1/ruta2/pagina.html#seccion
hash	seccion
port	-
protocol	http:
query	-

Métodos de `location`

`assign(url)`	Establece la URL de la página Es equivalente a **`location.href=url`**
`replace(url)`	Establece la URL de la página Similar a **`assign`**(), salvo que se borra la página del historial del navegador
`reload(bool)`	Recarga la página. Si el argumente es **`true`** se carga desde el servidor, de lo contrario se carga desde la caché del navegador.

Objeto navigator

- Contiene información sobre el navegador que estamos usando cuando abrimos una URL o documento local
- Habitualmente se emplea para:
  - Detectar tipo/versión del navegador
  - Comprobar si las cookies están habilitadas

Propiedades de navigator

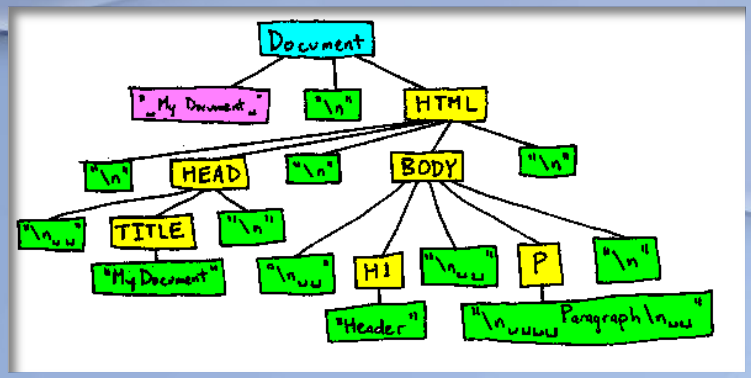
appCodeName	Cadena que representa el nombre del navegador
appName	Cadena que representa el nombre oficial del navegador
appVersion	Cadena que representa la versión del navegador
cookieEnabled	Indica si las cookies están habilitadas
platform	Cadena que representa la plataforma sobre la que se ejecuta el navegador
userAgent	Cadena que el navegador usa para identificarse en servidores (navegador + versión)
javaEnabled	Boolean que indica si Java está habilitado

Objeto screen

- Se usa para obtener información sobre la pantalla del usuario

Propiedades de screen

availHeight **availWidth**	Altura/Anchura total para las ventanas Tiene en cuenta elementos del S.O. como la barra de tareas o los anchos de pantalla
colorDepth	Profundidad de color
height **Width**	Altura/anchura total de la pantalla



Introducción al DOM (*Document Object Model*)

Indice

UD3: Objetos predefinidos en JS

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 y realizar operaciones de:
  - Lectura
  - Modificación
  - Eliminación de elementos
  - Inserción de elementos

Objeto `document`

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

Colecciones de `document`

`anchors[]`	Array con todos los enlaces del documento
`forms[]`	Array con todos los formularios del documento
`images[]`	Array con todas las imágenes del documento
`links[]`	Array con todos los enlaces del documento

Propiedades de `document`

`lastModified`	Fecha de la última modificación de la página
`cookie`	Cookies del documento
`domain`	Nombre de dominio del servidor que envió el documento
`referer`	URL desde la cual llegamos al documento actual (página anterior en el historial)
`title`	Título del documento (lectura/escritura)
`url`	URL completa del documento (lectura/escritura)

Métodos de `document`

Recomendadas

`getElementById(id)`	Devuelve el elemento con id **`id`**
`getElementsByClassName(class)`	Devuelve un array de elementos que tienen clase **`class`**
`getElementsByTagName(tagname)`	Devuelve un array de elementos cuya etiqueta es **`tagname`**
`querySelector(query)`	Devuelve el primer elemento que concuerda con el grupo de selectores especificados entre paréntesis
`querySelectorAll(query)`	Devuelve un array de elementos que concuerdan con el grupo de selectores especificados entre paréntesis

Métodos de `document`

Se desaconsejan, más ineficientes

`open()`	Abre el flujo de escritura para poder usar **`write()`**
`close()`	Cierra el flujo de escritura abierto con **`open()`**
`write(cadena)`	Permite escribir expresiones HTML dentro de un documento.
`writeln(cadena)`	Permite escribir expresiones HTML dentro de un documento, añadiendo 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

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]`
    - Nombre/id del marco
      - `[window].frames["frameidzo"]`
    - Atributo name de `<frame>`
      - `frameizdo`

EJERCICIO PROPUESTO

- Crea un programa que cuente el número de enlaces, párrafos e imágenes de la página
- Crea un programa que cuente el número de enlaces cuya URL contiene la palabra “linares”.
- Crea un programa que cuente el número de imágenes del tercer párrafo
- Crea un script que compruebe si hay alguna imagen con el atributo alt vacío.