Tema 3: Utilización de los objetos predefinidos del lenguaje

Desarrollo Web en Entorno Cliente - 2º DAW

Juana Mª Rodríguez García

1. INTRODUCCIÓN

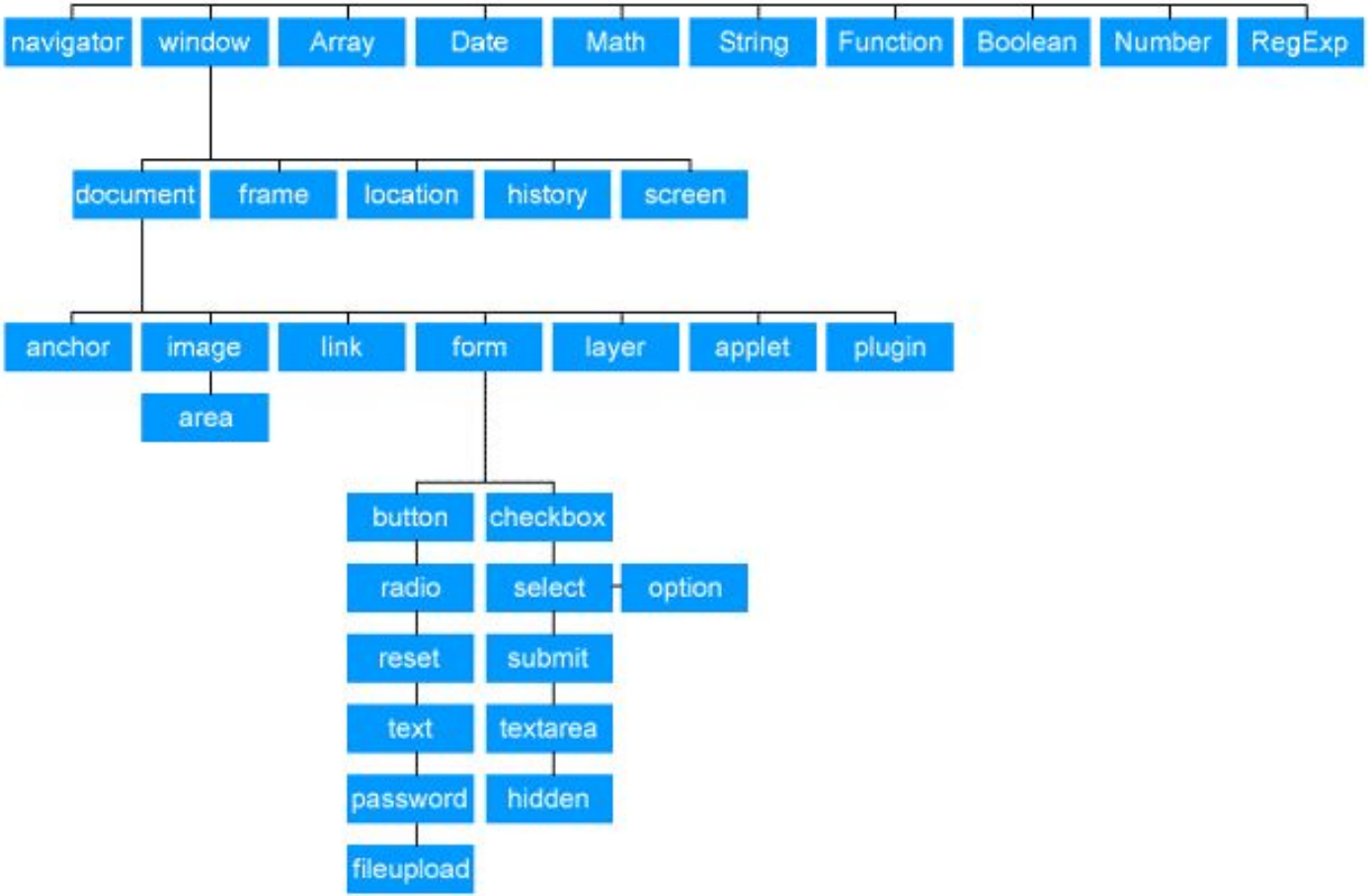
El tipo de datos fundamental de JavaScript es el **objeto**. Un objeto es un dato complejo, una colección *no ordenada* de propiedades cada una formada por un *nombre* y un *valor*. Este valor es un tipo de dato primitivo o un objeto.

En JavaScript todo lo que no es un string, un número, `true`, `false`, `null` o `undefined` es un objeto.

Podemos distinguir tres categorías o clases de objetos:

- **Objetos nativos** definidos por la propia especificación JavaScript (ECMAScript). En este tema nos centraremos en ellos
- **Objetos de plataforma,** son aquellos que pone a nuestra disposición el entorno de ejecución como el navegador
- **Objetos de usuario**, que son aquellos que se crean por programa durante la ejecución de código JavaScript. Veremos cómo crearlos en el próximo tema

JERARQUÍA DE OBJETOS JAVASCRIPT



https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=858:jerarquia-de-objetos-javascript-forms-elements-images-navigator-useragent-geolocation-online-cu01170e&catid=78&Itemid=206

1. INTRODUCCIÓN

Un objeto es una colección de :

- Propiedades (valores)
- Métodos (funciones)

Para acceder a una propiedad o método usamos el (.)

Objeto.propiedad;
Objeto.método(argumento);

2. OBJETOS NATIVOS O PREDEFINIDOS

Los objetos nativos NO dependen del navegador:

- ~~Number~~
- ~~String~~
- ~~Boolean~~
- **Date**
- **Math**
- Array
- Function
- RegExp

Para crear un objeto se usa la sintaxis:

```
let miObjeto= new Objeto();
```

2. OBJETOS NATIVOS O PREDEFINIDOS

Como vimos se puede crear un número

- usando su tipo primitivo :
  - (const num = 5)
- usando su objeto
  - (const num = new Number(5))

- Es mucho **más eficiente usar los tipos primitivos**. Aunque usemos tipo primitivo se considera un objeto y tenemos acceso a todas sus propiedades y métodos (num.toFixed(2)).

Ya hemos visto las principales propiedades y métodos de Number, String, Boolean y en este tema vamos a ver las de Math y Date.

2. OBJETOS NATIVOS O PREDEFINIDOS: DATE

Es la clase que usaremos siempre que vayamos a trabajar con fechas. Al crear una instancia de la clase le pasamos la fecha que queremos crear o lo dejamos en blanco para que nos cree la fecha actual. Si le pasamos la fecha podemos pasarle:

- let d= new Date ();
- let d = new Date (milisegundos);
- let d = new Date (cadena de fecha en formato AAAA-MM-DD o MM-DD-AAAA);
- let d = new Date (año, mes, dia, hora, minutos, segundos, milisegundos);
  //El mes comienza en 0, Enero sería 0

2. OBJETOS NATIVOS O PREDEFINIDOS: DATE

```
const date1=new Date()    // (es cuando he ejecutado la instrucción)

const date2=new Date(1532908800000)    // Mon Jul 30 2018 00:00:00 GMT+0200 (CEST) (miliseg. desde 1/1/1070)

const date3=new Date('2024-09-25')    //(la fecha pasada a las 0h. GMT)

const date4=new Date('2018-07-30 05:30')  // Mon Jul 30 2018 05:30:00 GMT+0200 (CEST) (la fecha pasada a las 05:300h. local)

const date5=new Date('07-30-2024')    // (OJO: formato MM-DD-AAAA)

const date6=new Date('30-07-2014')    // Invalid date

const date7=new Date('30-Jul-2024')    // (tb. podemos poner 'July')

const date8=new Date(2018,7,30)    // OJO: Thu Ago 30 2018 00:00:00 GMT+0200 (CEST) (OJO: 0->Ene,1->Feb... y a las 0h. local)

const date=new Date(2024,7,30,5,30)    // OJO: Thu Ago 30 2024 05:30:00 GMT+0200 (CEST) (OJO: 0->Ene,1->Feb,...)
```

2. OBJETOS NATIVOS O PREDEFINIDOS: DATE

- Cuando ponemos la fecha como texto, como separador de las fechas podemos usar `-`, `/` o `espacio`.
- Como separador de las horas debemos usar `:`.
- Cuando ponemos la fecha con parámetros numéricos (separados por `,`) podemos poner valores fuera de rango que se sumarán al valor anterior. Por ejemplo:

```
const date=new Date(2018,7,41)    // Mon Sep 10 2018 00:00:00 GMT+0200 (CEST) -> 41=31Ago+10
const date=new Date(2018,7,0)     // Tue Jul 31 2018 00:00:00 GMT+0200 (CEST) -> 0=0Ago=31Jul (el anterior)
const date=new Date(2018,7,-1)    // Mon Jul 30 2018 00:00:00 GMT+0200 (CEST) -> -1=0Ago-1=31Jul-1=30Jul
```

2. OBJETOS NATIVOS O PREDEFINIDOS: DATE

- Tenemos métodos *getter* y *setter* para obtener o cambiar los valores de una fecha:
  - **fullYear**: permite ver (*get*) y cambiar (*set*) el año de la fecha:
  - **month**: devuelve/cambia el número de mes, pero recuerda que 0->Ene,…,11->Dic
  - **date**: devuelve/cambia el número de día

```
const fecha=new Date('2018-07-30')
console.log( fecha.getFullYear() )
fecha.setFullYear(2019)
```

```
const fecha=new Date('2018-07-30')
console.log( fecha.getMonth() )
fecha.setMonth(8)
```

```
const fecha=new Date('2018-07-30')
console.log( fecha.getDate() )
fecha.setDate(-2)
```

2. OBJETOS NATIVOS O PREDEFINIDOS: DATE

- Tenemos métodos *getter* y *setter* para obtener o cambiar los valores de una fecha:
  - **day**: devuelve el número de día de la semana (0->Dom, 1->Lun, …, 6->Sáb). Este método NO tiene *setter*:
  - **hours, minutes, seconds, milliseconds**, : devuelve/cambia el número de la hora, minuto, segundo o milisegundo, respectivamente.
  - **time**: devuelve/cambia el número de milisegundos desde Epoch (1/1/1970 00:00:00 GMT):

```
const fecha=new Date('2018-07-30')
console.log( fecha.getDay() )
```

```
const fecha=new Date('2018-07-30')
console.log( fecha.getTime() )
fecha.setTime(1000*60*60*24*25)
```

2. OBJETOS NATIVOS O PREDEFINIDOS: DATE

Para mostrar la fecha tenemos varios métodos diferentes:

- .toString(): “Mon Jul 30 2018 02:00:00 GMT+0200 (CEST)”
- .toUTCString(): “Mon, 30 Jul 2018 00:00:00 GMT”
- .toDateString(): “Mon, 30 Jul 2018”
- .toTimeString(): “02:00:00 GMT+0200 (hora de verano de Europa central)”2024
- .toISOString(): “2018-07-30T00:00:00.000Z”
- .toLocaleString(): “30/7/2018 2:00:00”
- .toLocaleDateString(): “30/7/2018”
- .toLocaleTimeString(): “2:00:00”

2. OBJETOS NATIVOS O PREDEFINIDOS: DATE

NOTA: recuerda que las fechas son objetos y que se copian y se pasan como parámetro por referencia:

```
const fecha=new Date('2018-07-30')    // Mon Jul 30 2018 02:00:00 GMT+0200 (CEST)
const otraFecha=fecha
otraFecha.setDate(28)                 // Thu Jul 28 2018 02:00:00 GMT+0200 (CEST)
console.log( fecha.getDate() )        // imprime 28 porque fecha y otraFecha son el mismo objeto
```

Una forma sencilla de copiar una fecha es crear una nueva pasándole la que queremos copiar:

```
const fecha=new Date('2018-07-30')    // Mon Jul 30 2018 02:00:00 GMT+0200 (CEST)
const otraFecha=new Date(fecha)
otraFecha.setDate(28)                 // Thu Jul 28 2018 02:00:00 GMT+0200 (CEST)
```

2. OBJETOS NATIVOS O PREDEFINIDOS: DATE

- La comparación entre fechas funciona correctamente con los operadores `>`, `>=`, `<` y `<=`
- pero NO con `==`, `===`, `!=` y `!==` ya que compara los objetos y ve que son objetos diferentes.

Si queremos saber si 2 fechas son iguales (siendo diferentes objetos) el código que pondremos

- NO es `fecha1 === fecha2`
- sino `fecha1.getTime() === fecha2.getTime()`.

EJERCICIO 5: comprueba si es mayor tu fecha de nacimiento o el 1 de enero de este año

2. OBJETOS NATIVOS O PREDEFINIDOS: MATH

- Objeto que proporciona constantes y funciones matemáticas. Al contrario que el objeto Date **no podemos crear nuevos objetos** `Math`, existe un único objeto que podemos utilizar para realizar las operaciones matemáticas.
  - constantes: `.PI` (número pi), `.E` (número de Euler), `.LN2` (logaritmo natural en base 2), `.LN10` (logaritmo natural en base 10), `.LOG2E` (logaritmo de E en base 2), `.LOG10E` (logaritmo de E en base 10), `.SQRT2` (raíz cuadrada de 2), `.SQRT1_2` (raíz cuadrada de 1⁄2). Ejemplos:

```
console.log( Math.PI )              // imprime 3.141592653589793

console.log( Math.SQRT2 )           // imprime 1.4142135623730951
```

2. OBJETOS NATIVOS O PREDEFINIDOS: MATH

- `Math.round(x)`: redondea x al entero más cercano
- `Math.floor(x)`: redondea x hacia abajo (5.99 → 5. Quita la parte decimal)
- `Math.ceil(x)`: redondea x hacia arriba (5.01 → 6)
- `Math.min(x1,x2,...)`: devuelve el número más bajo de los argumentos que se le pasan.
- `Math.max(x1,x2,...)`: devuelve el número más alto de los argumentos que se le pasan.
- `Math.pow(x, y)`: devuelve x y (x elevado a y).
- `Math.abs(x)`: devuelve el valor absoluto de x.
- `Math.random()`: devuelve un número decimal aleatorio entre 0 (incluido) y 1 (no incluido). Si queremos un número entre otros rangos haremos `Math.random() * (max - min) + min` o si lo queremos sin decimales `Math.round(Math.random() * (max - min) + min)`
- `Math.cos(x)`: devuelve el coseno de x (en radianes).
- `Math.sin(x)`: devuelve el seno de x.
- `Math.tan(x)`: devuelve la tangente de x.
- `Math.sqrt(x)`: devuelve la raíz cuadrada de x

2. OBJETOS NATIVOS O PREDEFINIDOS: 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.

2. OBJETOS NATIVOS O PREDEFINIDOS: STRING

Propiedad	Descripción
length	Contiene la longitud de una cadena.

Método	Descripción
charAt()	Devuelve el carácter especificado por la posición que se indica entre paréntesis.
charCodeAt()	Devuelve el Unicode del carácter especificado por la posición que se indica entre paréntesis.
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.

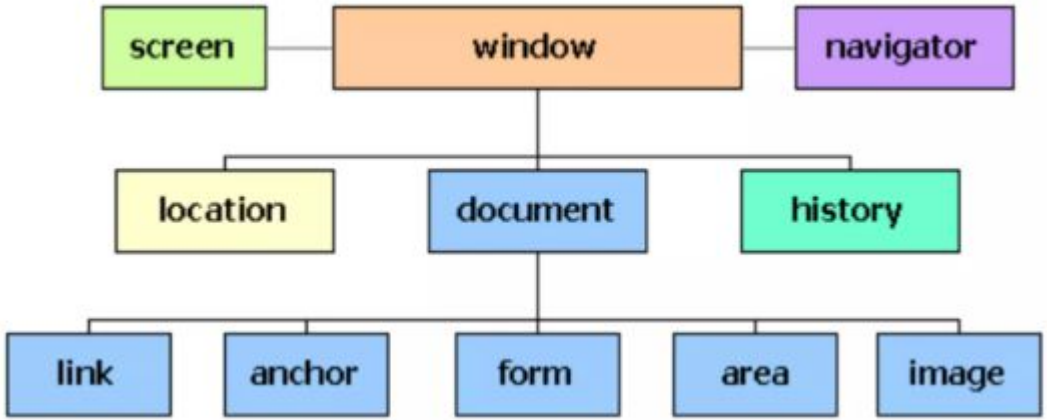
2. OBJETOS NATIVOS O PREDEFINIDOS: STRING

Método	Descripción
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.

2. OBJETOS NATIVOS O PREDEFINIDOS: BOOLEAN

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

3. OBJETOS DE PLATAFORMA O DE ALTO NIVEL



3. OBJETOS DE ALTO NIVEL: WINDOWS

Acceso a propiedades y métodos:

- **window.nombrePropiedad**
- **window.nombreMetodo(parámetros)**

Como la ventana es el objeto principal, podemos omitir su nombre y usar directamente:

- **nombrePropiedad**
- **nombreMetodo(parámetros)**

3. OBJETOS DE ALTO NIVEL: WINDOWS

- El objeto Window es un objeto que solo está presente al trabajar con Javascript en navegadores, no estando presente en entornos como NodeJS.

- El motivo de que este objeto sólo exista en navegadores es que fuera de ellos carece de sentido, ya que posee propiedades y controla elementos relacionado con lo que ocurre en la “ventana” del navegador.

- Los métodos que estudiamos en el tema anterior cómo **alert, prompt,** etc. forman parte del objeto Window. Para hacer llamada a estos métodos no hace falta nombrar explícitamente Window (el navegador ya se encarga de ello).

3. OBJETOS DE ALTO NIVEL: WINDOWS

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).

3. OBJETOS DE ALTO NIVEL: WINDOWS

Propiedad	Descripción
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 / window	Devuelve la ventana actual.
status	Ajusta el texto de la barra de estado de una ventana.
top	Nombre alternativo de la ventana del nivel superior

3. OBJETOS DE ALTO NIVEL: WINDOWS

Método	Descripción
alert(mensaje)	Muestra una ventana emergente de alerta y un botón de aceptar.
blur()	Elimina el foco de la ventana actual.
clearInterval(id)	Resetea el cronómetro ajustado con setInterval().
setInterval(expresion,tiempo)	Llama a una función o evalúa una expresión en un intervalo especificado (en milisegundos).
clearTimeOut(nombre)	Cancela el intervalo referenciado por 'nombre'.
setTimeOut(expresion,tiempo)	Evalua la expresión después de que hayan pasado un intervalo de tiempo (en milisegundos).
close()	Cierra la ventana actual.
confirm(mensaje)	Muestra una ventana emergente con un mensaje, un botón de aceptar y un botón de cancelar.

3. OBJETOS DE ALTO NIVEL: WINDOWS

Método	Descripción
focus()	Coloca el foco en la ventana actual.
moveBy(x,y)	Mueve la ventana actual el número de píxeles especificados.
moveTo(x,y)	Mueve la ventana a la posición especificada.
open(url,nombre,características)	Abre una nueva ventana de navegación.
prompt(mensaje,respuesta_defecto)	Muestra una ventana de diálogo para introducir datos.
scrollBy(x,y)	Mueve el scroll de la ventana el número de píxeles indicados.
scrollTo(x,y)	Mueve el scroll de la ventana a la posición indicada.

3. OBJETOS DE ALTO NIVEL: WINDOWS

```
// Creamos un intervalo que cada 15 segundos muestra mensaje hola
let idA=setInterval("alert('hola');",15000);
// Creamos un timeout que cuando pasen 3 segundos muestra mensaje adios
let idB=setTimeout("alert('adios');",3000);
// Creamos un timeout que cuando pasen 5 segundos muestra mensaje estonoseve
let idC=setTimeout("alert('estonoseve');",5000);
// Cancelamos el último timeout
clearTimeout(idC);
```

3. OBJETOS DE ALTO NIVEL: WINDOWS

Ejercicios:

- Redirige al usuario a Google después de 5 segundos
- Imprime en la consola la URL de la página
- Crea dos botones en HTML, uno que lleve a la página anterior (`Atrás`) y otro que lleve a la siguiente página (`Adelante`), usando el historial del navegador.
- Crea un botón que, al hacer clic, recargue la página actual.
- Cambia el título de la página a "Nueva Página" después de que el usuario haga clic en un botón.
- Imprime el ancho y el alto de la ventana del navegador
- Crea un botón que abra una nueva ventana con la página de Wikipedia y otro botón que cierre esa ventana.

3. OBJETOS DE ALTO NIVEL: WINDOWS

Método	Descripción
toolbar = [yes|no|1|0]	Indica si la ventana tendrá o no barra de herramientas (yes=1; no=0).
location = [yes|no|1|0]	Indica si la ventana tendrá campo de localización o no (yes=1; no=0).
directories = [yes|no|1|0]	Indica si la ventana tendrá botones de dirección o no (yes=1; no=0).
status = [yes|no|1|0]	Indica si la ventana tendrá barra de estado o no (yes=1; no=0).
menubar = [yes|no|1|0]	Indica si la ventana tendrá barra de menús o no (yes=1; no=0).
scrollbars = [yes|no|1|	Indica si la ventana tendrá barras de desplazamiento o no (yes=1; no=0).
resizable = [yes|no|1|0].	Indica si la ventana se podrá redimensionar o no (yes=1; no=0).
width = px / heigth = px	Ancho y alto de la ventana en píxeles
outerWidth = px / outerHeigth = px	Ancho y alto total de la ventana en píxeles.
top = px / left = px	Distancia de la ventana desde la parte superior e izquierda en píxeles.