



JS

UT3. OBJETOS PREDEFINIDOS EN JAVASCRIPT

Desarrollo Web en Entorno
Cliente
2ºDAW

1. OBJETOS NATIVOS DE JAVASCRIPT

JavaScript proporciona una serie de objetos definidos nativamente que no dependen del navegador

Para instanciar un nuevo objeto de alguna clase se utiliza la palabra clave `new`

- Ejemplo:

```
const objeto = new Object();
```

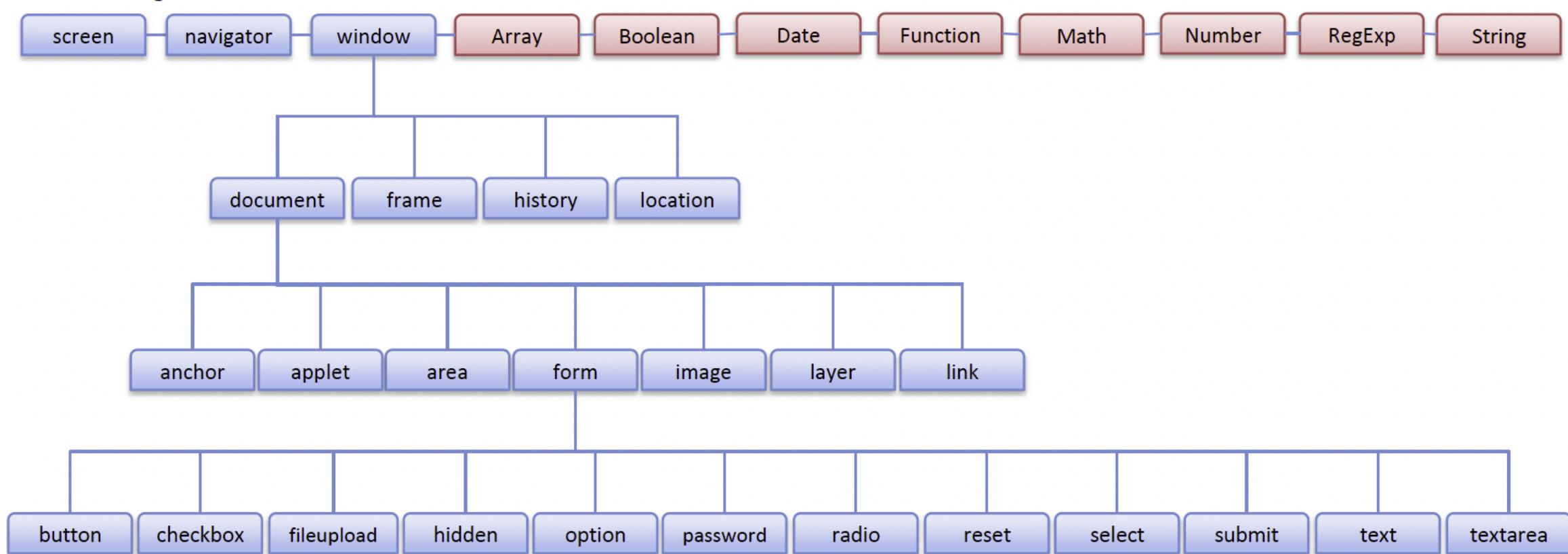
1. OBJETOS NATIVOS DE JAVASCRIPT

Para acceder a las propiedades y métodos de los objetos se hace con el operador punto ‘ . ’:

```
objeto.nombre_propiedad;  
objeto.nombre_funcion( [parametros] );
```

1. OBJETOS NATIVOS DE JAVASCRIPT

Los objetos nativos de JS se ordenan de forma jerárquica



1.2 EL OBJETO DATE

Permite trabajar con fechas y horas

Almacena el número de milisegundos desde las 00:00:00 UTC del 1 de enero de 1970

El constructor Date:

```
new Date()
new Date(milisegundos)
new Date(cadenaFecha)
new Date(año_num,mes_num,dia_num
        [,hor_num,min_num,seg_num,mils_num])
```

Si no proporciona argumentos, el constructor crea un objeto Date con la hora y fecha de hoy según la hora local.

Fuente: https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Date

1.2 EL OBJETO DATE.

Funciones del objeto global Date:

Métodos				
getDate()	getTime()	getUTCMonth()	setMonth()	setUTCMonth()
getDay()	getTimezoneOffset() set()	getUTCSeconds())	setSeconds()	setUTCSeconds()
getFullYear())	getUTCDate()	parse()	setTime()	toDateString()
getHours()	getUTCDay()	setDate()	setUTCDate()	toLocaleDateString())
getMilliseconds()	getUTCFullYear())	setFullYear()	setUTCFullYear())	toLocaleTimeString())
getMinutes()	getUTCHours()	setHours()	setUTCHours()	toLocaleString()
getMonth()	getUTCMilliseconds()	setMilliseconds()	setUTCMilliseconds())	toTimeString()
getSeconds()	getUTCMinutes())	setMinutes()	setUTCMinutes())	toUTCString()

1.3 EL OBJETO MATH

Permite realizar operaciones matemáticas complejas en JavaScript.

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Math

Métodos		
abs()	exp()	random()
acos()	floor()	round()
asin()	log()	sin()
atan()	max()	sqrt()
ceil()	min()	tan()
cos()	pow()	

Propiedades
E
LN2
LN10
LOG2E
LOG10E
PI
SQRT1_2
SQRT2

1.4 EL OBJETO NUMBER

Permite realizar operaciones con tipo de datos numérico.

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Number

Métodos
<code>toExponential()</code>
<code>toFixed()</code>
<code>toPrecision()</code>

Propiedades
<code>MAX_VALUE</code>
<code>MIN_VALUE</code>
<code>NaN</code>
<code>NEGATIVE_INFINITY</code>
<code>POSITIVE_INFINITY</code>

1.5 EL OBJETO STRING

Permite manejar cadenas de texto

Métodos				Propiedades
anchor()	fixed()	link()	strike()	Lenght
big()	fontcolor()	match()	sub()	
blink()	fontsize()	replace()	substr()	
bold()	fromCharCode())	search()	substring()	
charAt()	indexOf()	slice()	sup()	
charCodeAt()	italics()	small()	toLowerCase()	
concat()	lastIndexOf()	split()	toUpperCase()	

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/String

2. INTERACCIÓN DE OBJETOS CON EL NAVEGADOR

Además de los presentados anteriormente existen otro tipo de objetos que permiten manipular características del navegador en si mismo

- navigator
- screen
- window
- document
- history
- location

2.1 EL OBJETO NAVIGATOR

Permite identificar las características de la plataforma sobre la cual se ejecuta la aplicación web. Datos que podemos obtener:

- Tipo de navegador
- Versión del navegador
- Sistema operativo
- Geolocalización del dispositivo

<https://developer.mozilla.org/en-US/docs/Web/API/Navigator>

Obtener geolocalización

<https://developer.mozilla.org/es/docs/Web/API/Geolocation/getCurrentPosition>

2.2 EL OBJETO SCREEN

Corresponde a la pantalla del usuario.

Todas sus propiedades son de solo lectura.

Propiedades
availHeight
availWidth
colorDepth
height
pixelDepth
width

<https://developer.mozilla.org/en-US/docs/Web/API/Screen/availHeight>

2.3 EL OBJETO WINDOW

Se considera el objeto más importante de JavaScript

Permite gestionar las ventanas del navegador

Es un objeto implícito, con lo cual no es necesario nombrarlo para acceder a los objetos que se encuentran debajo de su nivel de jerarquía.

<https://developer.mozilla.org/es/docs/Web/API/Window>

2.3 EL OBJETO WINDOW

Métodos		
alert()	forward()	setinterval()
back()	home()	setTimeOut()
blur()	moveTo()	scrollBy()
close()	open()	scrollTo()
confirm()	print()	stop()
find()	prompt()	setinterval()
focus()	resizeTo()	setTimeOut()

Propiedades		
closed	location	pageYoffset
defaultStatus	locationbar	parent
document	menubar	personalbar
frames	name	scrollbars
history	opener	self
innerHeight	outerHeight	status
innerWidth	outerWidth	toolbar
length	pageXoffset	top

2.4 EL OBJETO DOCUMENT

Se refiere a la página web que está cargado en el navegador

Desde él podemos acceder y manipular todos los elementos de DOM

<https://developer.mozilla.org/es/docs/Web/API/Document>

2.5 EL OBJETO HISTORY

Almacena las referencias de las páginas web visitadas.

Las referencias se guardan en una lista utilizada principalmente para desplazarse entre dichas páginas web.

Métodos
<code>back()</code>
<code>forward()</code>
<code>go()</code>

Propiedades
<code>current</code>
<code>length</code>
<code>next</code>
<code>previous</code>

<https://developer.mozilla.org/en-US/docs/Web/API/History>

2.6 EL OBJETO LOCATION

Corresponde a la URL

Su principal función es la de consultar las diferentes partes que forman una URL como por ejemplo:

- El dominio.
- El protocolo.
- El puerto.

Métodos	Propiedades
assign()	hash
reload()	host
replace()	hostname
	href
	pathname
	port
	protocol
	search

<https://developer.mozilla.org/es/docs/Web/API/Location>

3. GENERACIÓN DE ELEMENTOS HTML DESDE EL CÓDIGO

Uno de los principales objetivos de JavaScript es convertir un documento HTML estático en una aplicación web dinámica.

Por ejemplo, es posible ejecutar instrucciones que crean nuevas ventanas con contenido propio, en lugar de mostrar dicho contenido en la ventana activa.

3.1 CREAR UN NUEVO ELEMENTO HTML

Podemos crear un nuevo elemento escribiendo directamente código HTML en el documento:

```
<body>
  ...
  <script>
    const so = navigator.platform;
    document.write(`<h1>Está navegando sobre el SO: ${so}</h1>`)
  </script>
</body>
</html>
```

3.1 CREAR UN NUEVO ELEMENTO HTML

También podemos crear código HTML en ventanas emergentes:

```
<body>

    <script>
        const texto = prompt("Ingresa un título para la nueva ventana: ");
        const ventanaNueva= window.open();
        ventanaNueva.document.write("<h1>" + texto
        + "</h1>");

    </script>
</body>
```

3.2 GENERAR NUEVO ELEMENTO CON FUNCIONES DE DOCUMENT

Podemos utilizar las siguientes funciones del objeto global document para crear nuevos elementos e insertarlos dentro de otros elementos:

- `document.createElement(tagname,[options])`
- `document.createTextNode(text)`
- `elemento.appendChild(elementoHijo)`

Ejemplo:

<https://developer.mozilla.org/es/docs/Web/API/Document/createElement>