I.E.S. PALOMERAS VALLECAS

OBJETOS PREDEFINIDOS EN JAVASCRIPT

UT3.- OBJETOS PREDEFINIDOS

Susana López Luengo

Objetos nativos JavaScript

- Identificar y conocer los objetos predefinidos del lenguaje.
- Comprender las propiedades y métodos de los principales objetos.
- Aprender a manipular un documento HTML desde sentencias JavaScript.
- Manipular y gestionar la creación y apariencia de las ventana del navegador y la comunicación entre ellas.

Objetos nativos JavaScript

- Algunas de las operaciones más comunes para las cuales se diseñó JavaScript son:
  - Abrir una nueva ventana en el navegador
  - Escribir un texto en un document
  - Redirigir un navegador a otra ubicación
  - Validar los datos de un formulario
  - Cambiar la página de un marco
  - Etc
- Ventana, documento, ubicación, formulario y marco se denominan objetos predefinidos en JavaScript.

Objetos nativos JavaScript

- Ventana, documento, ubicación, formulario y marco se denominan objetos predefinidos en JavaScript.
- Dichos objetos son elementos programables con los que podemos
  - manipular para cambiar sus propiedades,
  - realizar tareas a través de sus métodos
  - ejecutar un evento relacionado con el mismo objeto.

Objetos nativos JavaScript

- Las **propiedades** son las características de un objeto.
- Los **métodos** son funciones o tareas específicas que pueden realizar los objetos
- Los **eventos** son situaciones que pueden llegar a realizarse o no. Cada objeto puede reconocer una serie de eventos.
- JavaScript proporciona una serie de objetos definidos nativamente que no dependen del navegador
- Para crear un objeto se utiliza la palabra clave new.

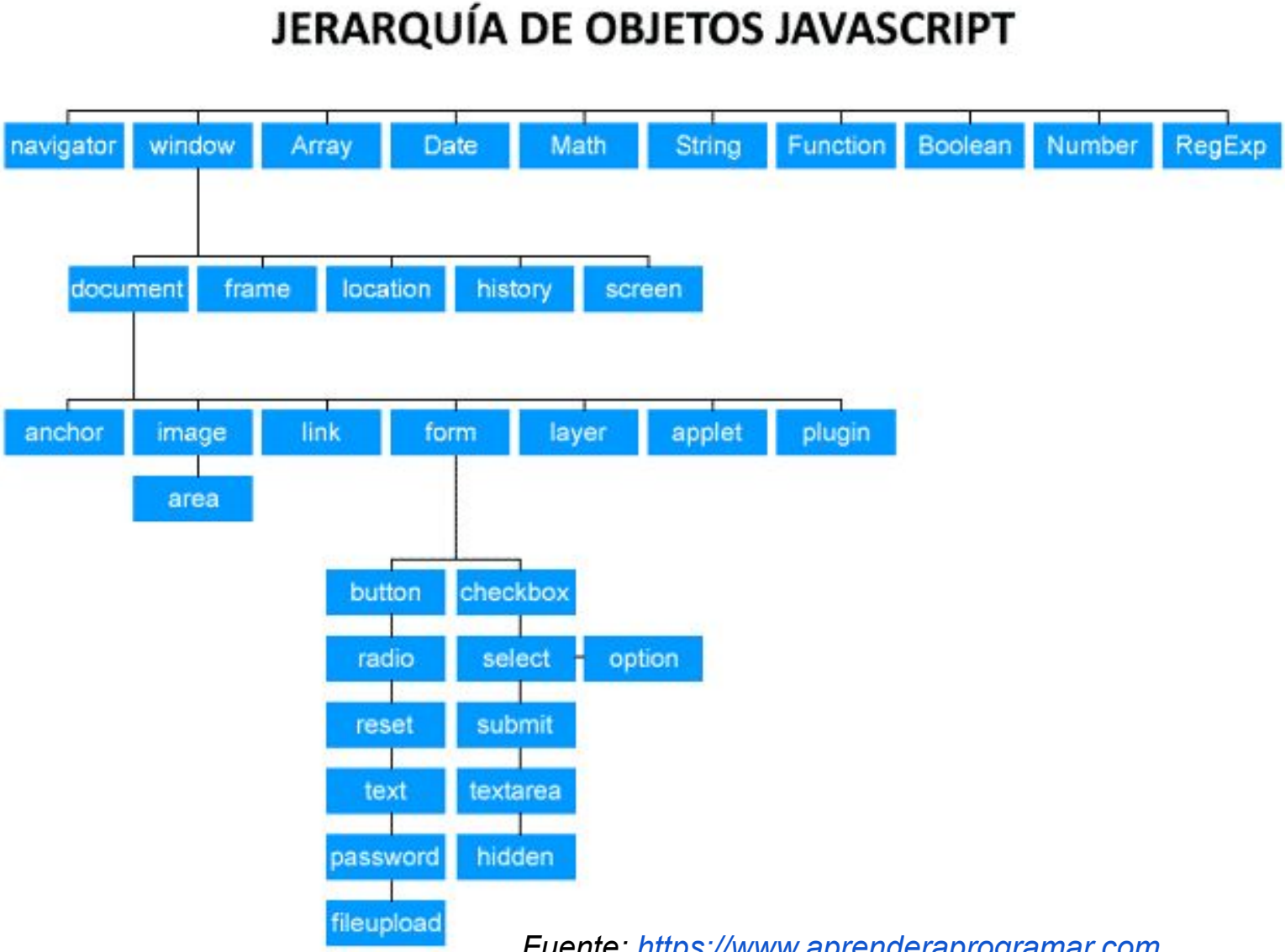
```
var miObjeto = new Object();
```

Objetos nativos JavaScript

- A la variable llamada miObjeto se le asigna una nueva instancia de un objeto que puede ser cualquiera de los objetos predefinidos de JavaScript o los definidos por el usuario.
- En JavaScript se accede a las propiedades y a los métodos de los objetos mediante el operador punto (“.”):

```
miObjeto.nombrePropiedad;
miObjecto.Funcion(parámetros);
```

Objetos nativos JavaScript



Fuente: https://www.aprenderaprogramar.com

Objetos nativos JavaScript

objeto Date

- Permite realizar controles relacionados con el tiempo en las aplicaciones web.
- Cuenta con una serie de métodos divididos en tres subconjuntos:
  - **Métodos de lectura**. Empiezan por el prefijo **get**.
  - **Métodos de escritura**. Empiezan por el prefijo **set**
  - **Métodos de conversión**. Convierte objetos de tipo Date en cadenas de texto o en milisegundos.

Objetos nativos JavaScript

objeto Date

- Las fechas en JavaScript se miden como el número de milisegundos transcurridos desde la época UNIX (1 de enero de 1970 00:00:00 UTC).
- Para crear un objeto de tipo Date:

```
var miFecha = new Date(milisegundosDesdeEpocaUNIX);
var miFecha = new
Date(año,mes,dia[,hora,minuto,segundo,milisegundo]);
```

- El año debe expresarse con 4 dígitos, el mes entre 0 y 11, y el día entre 1 y 31.
- Por defecto JavaScript crea un objeto Date con la hora local actual del sistema

```
var FechaActual = new Date();
```

Objetos nativos JavaScript

métodos de Date

- `getDate()` Obtiene el día como un número (1-31)
- `getDay()` Obtiene el día de la semana como un número (0-6)
- `getFullYear()` Obtiene el año con 4 dígitos (yyyy)
- `getHours()` Obtiene la hora(0-23)
- `getMilliseconds()` Obtiene los milisegundos (0-999)
- `getMinutes()` Obtiene los minutos (0-59)
- `getMonth()` Obtiene el mes (0-11)
- `getSeconds()` Obtiene los segundos (0-59)
- `getTime()` Obtiene la hora (en milisegundos desde el 1 de Enero de 1970) https://www.w3schools.com/js/js_date_methods_set.asp

Objetos nativos JavaScript

objeto Date

Ejemplo: Calcular el número de noches transcurrido desde la fecha de entrada y salida de un cliente en un hotel

```
var entrada = new Date(2019,7,1);
var salida = new Date(2019,7,6);
var noches = (salida.getTime()-
   entrada.getTime())/86400000;
console.log(noches);
console.log(entrada.toLocaleString());
console.log(entrada);
```

Objetos nativos JavaScript

objeto Math

- El objeto Math es la calculadora de JavaScript
- Podremos realizar operaciones aritméticas, raíces cuadradas, logaritmos, operaciones trigonométricas y, muy importante, obtener números pseudoaleatorios.
- Math no es un constructor, sino directamente un objeto
- Cómo en Java todos los métodos del objeto Math son estáticas

Objetos nativos JavaScript

algunos métodos del objeto Math

- abs(x) // Valor absoluto de x
- ceil(x) // Redondea hacia arriba
- floor(x) // Redondea hacia abajo
- max(x, y, z, ..., n) // Devuelve el valor más alto
- min(x, y, z, ..., n) // Devuelve el valor más bajo
- pow(x, y) // Devuelve x elevado a y
- random() // Devuelve un número aleatorio
- round(x) //Redondea un número
- sqrt(x) // raíz cuadrada de x

Objetos nativos JavaScript

objeto Math

Las propiedades de Math se utilizan para acceder a algunas constantes matemáticas de interés y son de sólo lectura.

- Math.E // Número de Euler (e)
- Math.PI // π
- Math.SQRT2 // raíz cuadrada de 2
- Math.LN2 // logaritmo neperiano de 2
- Math.LN10 // returns the natural logarithm of 10
- Math.LOG2E // returns base 2 logarithm of E
- Math.LOG10E // returns base 10 logarithm of E

https://www.w3schools.com/js/js_math.asp

Objetos nativos JavaScript

objeto Math

Ejemplo: Calcular el área de un círculo

```
<script type="text/javascript">
  var r = prompt("Introduce el radio:");
  var area = Math.PI * Math.pow(r, 2);
  alert("El Area del circulo es de: " + area + "
cms cuadrados");
</script>
```

Objetos nativos JavaScript

objeto Number

- Los métodos del objeto Number ayudan a trabajar con números
- Es poco común crear objetos de tipo Number con un contructor new.

```
numero =Number("123");
```

- Con JavaScript un valor primitivo numérico lo trata como un objeto Number, por lo que podemos acceder a métodos y propiedades.

```
var numero = 123;
console.log(numero.MAX_VALUE);
```

Objetos nativos JavaScript

métodos del objeto Number

- **toString([base])** // convierte un número en una cadena de texto. Con base se transforma a la base indicada.
  ```
  9.656.toString(); // devuelve "9.656"
  ```
- **toExponential()** // devuelve un String con el número redondeado y con notación exponencial
  ```
  9.656.toExponential(2); // devuelve 9.66e+0
  ```
- **toFixed(decimales)** // devuelve un String con los decimales indicados
  ```
  9.656.toFixed(2) // devuelve 9.66
  ```
- **toPrecision(precision)** // // devuelve un String con la precisión indicada
  ```
  9.656.toPrecision(2) // devuelve 9.7
  ```

Objetos nativos JavaScript

propiedades del objeto Number

- **MAX_VALUE** // Devuelve el valor más grande posible en JavaScript
- **MIN_VALUE** // Devuelve el valor más pequeño posible en JavaScript
- **NaN** // Representa un valor is Not a Number
- **NEGATIVE_INFINITY** // Representa a menos infinito
- **POSITIVE_INFINITY** // Representa a infinito

https://www.w3schools.com/js/js_number_methods.asp

Objetos nativos JavaScript

métodos de JavaScript para convertir números

- **Number(valor)** // convierte valor en un número

```
Number(true);          // devuelve 1
Number(new Date()); // devuelve 1404568027739
Number("10");           // devuelve 10
Number("10 20");       // devuelve NaN
```

- **parseFloat(valor)** // convierte valor en un número float

```
parseFloat("10.33"); // devuelve 10.33
```

- **parseInt(valor)** // convierte valor en un número entero

```
parseInt("10.33");    // devuelve 10
parseInt("10 años"); // devuelve 10
parseInt("años 10");  // devuelve NaN
```

Objetos nativos JavaScript. Number

parseInt(string[,base])

string se interpreta como un entero en la **base** indicada (por def 10)

```
parseInt('1010')    => 1010
parseInt('1010',2)  => 10

parseInt('12',8)    => 10
parseInt('10',10)   => 10
parseInt('a',16)    => 10
```

```
Number('60')        => 60
Number('01xx')      => NaN

parseInt('60')      => 60
parseInt('60.45')   => 60
parseInt('xx')      => NaN

parseInt('01xx')    => 1
parseInt('01+4')    => 1
```

```
(1).toFixed(2)          => "1.00"
(1).toPrecision(2)      => "1.0"

1.toFixed(2)            => Error

Math.PI.toFixed(4)      => "3.1416"
Math.E.toFixed(2)       => "2.72"

(1).toExponential(2)    => "1.00e+0"

(31).toString(2)        => "11111"
(31).toString(10)       => "31"
(31).toString(16)       => "1f"

(10.75).toString(2)     => "1010.11"
(10.75).toString(16)    => "a.c"
```

Objetos nativos JavaScript

objeto String

- Es un objeto que representa una serie de caracteres dentro de una cadena de comillas dobles o simples.

```
sObj = new String("prueba"); // crea un objeto String
```

- La función global String()también se puede llamar sin poner new delante para crear una cadena primitiva:

```
sPrim = String('prueba'); //crea una Cadena primitiva
```

- Las cadenas literales en el código fuente Javascript crean también cadenas:

```
sTbPrim = "prueba"; // crea Cadena primitiva
```

Objetos nativos JavaScript

métodos de búsqueda en objeto String

- **indexOf(valorDeBusqueda)** Devuelve el índice de la primera ocurrencia, si no devuelve -1
- **lastIndexOf(valorDeBusqueda)** Devuelve el índice de la última ocurrencia, si no devuelve -1
- **startsWith(cadena)** Indica si una cadena comienza por los caracteres indicados
- **endsWith(cadena)** Indica si una cadena termina por los caracteres indicados
- **search(expRegular)** Realiza una búsqueda de una expresión regular en una cadena especificada.
- **match(expRegular)** Se usa para buscar en base a una expresión regular dentro de una cadena.

Objetos nativos JavaScript

métodos para extraer partes de un objeto String

- **slice(comienzo,fin)** Extrae una sección de una cadena y devuelve una nueva cadena.
- **substr(comienzo,fin)** Devuelve los caracteres de una cadena comenzando en la localización especificada y hasta el número de caracteres especificado.
- **substring(comienzo,fin)** Devuelve los caracteres de una cadena entre dos índices dentro de la cadena.
- **split(separación)** Divide un objeto String en array de cadenas, separando la cadena en subcadenas, si se pasa "" devolverá un array con cada carácter

Objetos nativos JavaScript

métodos de un objeto String

- **charAt(posicion)** permite acceder a la letra de la posición de la cadena, tiene el mismo efecto que [posicion]

```
'gato'.charAt(1) // devuelve "a"
'gato'.[1] // devuelve "a"
```

  - No se recomienda utilizar corchetes con cadenas
- **charCodeAt(posición)** Devuelve el valor Unicode del carácter en el índice especificado.
- **concat(cadena2,cadena3,...)** Combina el texto de dos cadenas y devuelve una nueva cadena.

Objetos nativos JavaScript

más métodos del objeto String

- **replace()** Se usa para emparejar una expresión regular con una cadena, y reemplazar la subcadena emparejada con una nueva subcadena.
- **toLowerCase()** Devuelve el valor de la cadena que realiza la llamada en minúsculas.
- **toUpperCase()** Devuelve el valor de la cadena que realiza la llamada en mayúsculas.
- **trim()** Elimina los espacios en blanco al principio y al final de la cadena

Objetos nativos JavaScript

propiedades de un objeto String

- **length**, esta propiedad devuelve el número de caracteres que componen la cadena de caracteres

```
alert('juan'.length);
```

https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/String

https://www.w3schools.com/jsref/jsref_obj_string.asp

Objetos nativos JavaScript. String

```
jq — bash — 36×23
venus:~ jq$ node
> "La Ñ en ISO-8859-1 es: \xd1"
'La Ñ en ISO-8859-1 es: Ñ'
> "La á en ISO-8859-1 es: \xe1"
'La á en ISO-8859-1 es: á'
> "Backslash \\ debe escaparse"
'Backslash \\ debe escaparse'
> "El Euro: \u20ac"
'El Euro: €'
> "El Yen Japones: \xa5"
'El Yen Japones: ¥'
> "Ciudad"[1]
'i'
> "Ciudad".charCodeAt(1)
105
> String.fromCharCode(105)
'i'
> "Ciudad".substring(3,5)
'da'
> "Ciudad".substring(3,5).length
2
> .exit
venus:~ jq$
```

La Ñ existe en el código ISO-8859-1 y su código numérico en hexadecimal es d1, por lo que se puede incluir en un string tecleando Ñ o como \xd1.

La á existe también en el código ISO-8859-1 y la introducimos tecleando el acento y luego la letra a o con el código numérico en hexadecimal \xe1.

La barra inclinada (backslash) debe escaparse (\\) para que se visualice.

EL Euro no existe en ISO-8859-1 porque este código se creo antes de existir el Euro. Unicode se actualizó al aparecer el Euro añadiendo el símbolo € con el código \u20ac.

EL Yen Japonés si existe en ISO-8859-1: código hexadecimal \xa5.

Un string es un array (matriz) de caracteres, numerados de 1 a n-1 (último-1). **"Ciudad"[1]** indexa el segundo carácter del string, el primero será **"Ciudad"[0]**.

Al invocar el método **charCodeAt(1)** con el operador **"."** sobre el string **"Ciudad"** nos devuelve el valor numérico decimal del **punto del código** del 2º carácter ("i").

String.fromCharCode(105) realiza la operación inversa, devuelve un string con el carácter cuyo valor (punto del código) se pasa como parámetro.

"Ciudad".substring(3,5) devuelve la subcadena entre 3 y 5: **"da"**

"Ciudad".substring(3,5).length devuelve la longitud de la subcadena devuelta ("da"), que tiene 2 caracteres.

Objetos nativos JavaScript. String

Métodos de encapsulado HTML de objeto String

- Estos métodos devuelven una copia de la cadena encapsulada dentro de una etiqueta HTML
  - Por ejemplo, "test".bold() devuelve "test"

anchor()	italics()
big()	link()
blink()	small()
bold()	strike()
fixed()	sub()
fontcolor()	sup()
fontsize()	

Estos métodos no respetan los estándares de la W3C. Por este motivo es importante prestar mucha atención a su uso .

En muchos casos es preferible el uso de un diseño basado en hojas de estilo en cascada.(CSS)

Práctica guiada 1-1

UT03-Pr02ObjetosPredefinidos hasta ej 4

WEBGRAFÍA Y ENLACES DE INTERÉS

- https://www.w3schools.com/js/
- Curso Desarrollo de aplicaciones con HTML,node.js y Javascript. UPM. Miriadax
- https://www.aprenderaprogramar.com


ESCUELA PÚBLICA:
DE TOD@S
PARA TOD@S
Vallecas