3. OBJETOS PREDEFINIDOS DE JAVASCRIPT

Desarrollo Web en entorno cliente

IES Clara del Rey

Funciones predefinidas

- **parseInt(cadena)**: convierte una cadena de texto entero a entero .
- **parseFloat(cadena)**: convierte una cadena de texto en decimal.
- **isNaN(cadena)**: NaN es la abreviación de “Not a Number”. Esta función comprueba si una cadena de caracteres puede ser considerada un número (false) o no (true).

Qué son los objetos

Javascript no es un lenguaje de programación orientado a objetos puro porque, aunque utiliza objetos frecuentemente.

En JS se usan objetos y no tanto se programa orientado a objetos.

Los objetos son una herramienta de lenguajes de programación en la que se unen:

- **Propiedades**: se refieren a los datos
- **Métodos**: se refieren a la funcionalidad

Acceso:

- **Propiedades**: miObjeto.miPropiedad
- **Métodos**: miObjeto.miMetodo(parametro1,parametro2) o miObjeto.miMetodo()

Objetos predefinidos de Javascript

Las ***clases*** que se encuentran disponibles de manera nativa en Javascript

- **String**, para el trabajo con cadenas de caracteres.
- **Date**, para el trabajo con fechas.
- **Math**, para realizar funciones matemáticas.
- **Number**, para realizar algunas cosas con números
- **Boolean**, trabajo con boleanos.

Las clases se escriben con la primera letra en mayúsculas

- **Array**:
- **Function**
- **RegExp**

Diferencias de nomenclatura de clases y objetos

Las clases son descripciones de la forma y funcionamiento de los objetos.

Con las clases generalmente no se realiza ningún trabajo, sino que se hace con los objetos, que creamos a partir de las clases.

- **Number**: clase
- **RegExp**: clase compuesta por dos palabras
- **miCadena**: objeto de la clase String (dos palabras)
- **fecha**: objeto de la clase Date
- **miFunción()**: función definida por el desarrollador

Cadenas de caracteres: String

En javascript las variables de tipo texto son objetos de la clase String.

Para crear un objeto de la clase String sólo hay que asignar un texto (entrecomillado) a una variable.

Propiedades

length: número de caracteres del String

Métodos

- **charAt(indice)**
- **indexOf(carácter,desde)**
- **lastIndexOf(carácter,desde)**
- **replace(substring_a_buscar,nuevoStr)**

Cadenas de caracteres: String

Métodos (cont)

- **split(separador)**
- **substring(inicio,fin)**
- **slice(inicio,fin)**
- **toLowerCase() / toUpperCase()**
- **toString()**

- Otros métodos de String

Manejo de fechas y horas: Date

Para trabajar con fechas necesitamos instanciar un objeto de la clase **Date**.

```
let miFecha = new Date()  // día y hora actuales
let miFecha = new Date(año,mes,dia,hora,minutos,segundos)
let miFecha2 = new Date(año,mes,dia)
```

Los meses en Date se numeran del 0 al 11 (siendo el 0 Enero y el 11 Diciembre)

Los formatos de entrada y salida de fechas y horas los encuentras **aqui**

Métodos de Date

- getDate() / setDate()
- getDay()
- getHours() / setHours()
- getMinutes() / setMinutes()
- getMonth() / setMonth()
- getSeconds() / setSeconds()
- getTime() / setTime()
- getFullYear() / setFullYear()
- toDateString() / toGMTString() / toUTCString()

Cálculos matemáticos: Math

Proporciona los mecanismos para realizar operaciones matemáticas en Javascript.

Las propiedades y métodos de la clase Math son propiedades y métodos de clase. Para utilizarlos se opera a través de la clase en lugar de los objetos.

Constantes

- E: Número E o constante de Euler
- LN10: Logaritmo neperiano de 10
- PI: Número pi
- SQRT2: Raiz cuadrada de 2

Cálculos matemáticos: Math

Métodos

- abs() / trunc()
- **Redondeo**: ceil() / floor() / round()
- **Trigonométricas**: acos() / asin() / atan() / cos() / sin() / tan()
- **Potencias**: exp() / pow() / sqrt()
- **Comparativas**: max() / min()
- **Aleatorio**: random()

Referencia de la clase Math

Tipos de datos numéricos: Number

Sirve para crear objetos que tienen datos numéricos como valor y realizar algunas operaciones sencillas.

Permite crear valores numéricos a partir de valores en otros tipos de datos.

```
var n1 = new Number()  //muestra un 0
var n2 = new Number("hola")  //muestra NaN
var n3 = new Number("123") //muestra 123
var n4 = new Number("123asdfQWERTY") //muestra NaN
var n5 = new Number(123456) //muestra 123456
var n6 = new Number(false) //muestra 0
var n7 = new Number(true) //muestra 1
```

Propiedades y métodos de Number

Propiedades

- NaN
- MAX_VALUE y MIN_VALUE (notación científica)
- NEGATIVE_INFINITY y POSITIVE_INFINITY
- Number.MAX_SAFE_INTEGER
  Number.MIN_SAFE_INTEGER (entero común)
- Number.EPSILON

Métodos

- Number.isNaN()
- Number.isFinite()
- Number.isInteger()
- Number.isSafeInteger()
- Number.parseFloat()
- Number.parseInt()

Valores booleanos desde otros valores: Boolean

Permite conseguir valores boleanos a partir de datos de cualquier otro tipo.

```
var b1 = new Boolean() //muestra false
var b2 = new Boolean("") //muestra false
var b25 = new Boolean(false) //muestra false
var b3 = new Boolean(0) //muestra false

var b35 = new Boolean("0") //muestra true
var b4 = new Boolean(3) //muestra true
var b5 = new Boolean("Hola") //muestra true
```

La ventana del navegador: Window

Objeto que solo está presente al trabajar con Javascript en navegadores (no es válido para NodeJS).

Engloba propiedades y elementos relacionados con lo que ocurre en la "ventana" del navegador. No es necesario nombrar el objeto al invocar sus propiedades o métodos.

Propiedades

- document
- history
- innerHeight
- innerWidth
- location
- name
- ...

Referencia del objeto Window

El objeto Window

Métodos

- alert(texto)
- prompt(pregunta,inicializacion_de_la_respuesta)
- confirm(texto)
- back() / forward() / home()
- focus() / blur()
- open()
- stop()
- scrollTo(pixelsX,pixelsY)
- ...

Manejo de tiempos con Window

setInterval()

Llama a una función cada intervalo de tiempo, hasta que se llama a la función de parada *clearInterval*
myInterval = setInterval(function, milliseconds);
clearInterval(myInterval);

```
const myInterval = setInterval(myTimer, 1000);
function myTimer() {
  const date = new Date();
  document.getElementById("demo").innerHTML = date.getTime
}
function myStopFunction() {
  clearInterval(myInterval);
}
```

Manejo de tiempos con Window

setTimeout()

Llama a una función tras un tiempo especificado. *clearTimeout* evita la llamada a la función.
myTimeout = setTimeout(function, milliseconds);
clearTimeout(myTimeout)

```
let timeout;
function myFunction() {
  timeout = setTimeout(alertFunc, 3000);
}
function alertFunc() {
  alert("Hello!");
}
```

A por ello!


if ($(window).scrollTop() > header1_initialDistance) {
if (parseInt(header1.css('padding-top'), 10)
header1.css('padding-top', '' +
} else {
header1.css('padding-top', '' + header1_initialPadding
}
if ($(window).scrollTop() > header2_initialDistance) {
if (parseInt(header2.css('padding-top'), 10)
header2.css('padding-top',