**DWEC Apuntes  
*JavaScript* *tipos* de *Inserción***

<script type="text/javascript" src="rutaScript.js"></script>**-** external  
<script type=”text/javascript” > codigo </script> - Interno ***DOM***

document.getElementById(id); - devuelve el elemento.  
document.getElementByName(name); – devuelve un array de los que tienen el nombre  
document.getElementByTagName(nombre Elemento); – devuelve un array de los elementos  
document.getElementById(id).value; - valor del elemento

document.getElementByName(nombre)[0].value; - devuelve el valor del primer elemento que tiene el nombre.

document.getElementByName(nombre)[0].checked; - devuelve un booleano si está seleccionado

***Inicialización* *de* *variables***

var – local/global

let – local/global

const – constante

***Funciones***

function nombreFuncion(pm1,pm2){}

document.write(“elemento”); - Función para escribir

typeOf(var);- devuelve el tipo de dato del variable.

let valor=prompt(‘msj’) – El usuario introduce un valor al prompt, que se puede pasar a un variable.

***Numero*:**

parseInt(str); – cambia un string numero a un numero de verdad

x.toString();- cambia x a un string.

Math.floor();- redondea al numero más bajo: 0.9 = 0

Math.random();

Math.floor(Math.random()\*10) – devuelve un number random entre 0 a 9

Math.floor(Math.random()\*10)+1 – devuelve un number random entre 1 a 10

Math.floor(Math.random() \* (max - min) ) + min; - devuelve un numero entre min y max (max excluida)

Math.floor(Math.random() \* (max – min +1) ) + min; - devuelve un numero entre min y max ambos incluidos.

***String* *Methods*:**

Str.length(); – longitud de un String.

Str.slice (start,end);– devuelve un nuevo string con la parte cortada.

Str.substr(start,end); – Mismo que slice.

Str.replace(“1”,”2”); – el 2º parámetro reemplaza el 1º parámetro en una string. (case sensitive).

Str.toUpperCase(); - String a mayúscula.

Str.toLowerCase(); - String a minúscula.

Str.concat(str2); - String une str con str2.

Str.trim(); - quita los espacios blancos.

Str.trimStart(); - quita los espacios desde pos 0.

Str.trimEnd(); - quita espacios desde el final.

Str.charAt(pos); – coge la letra en posición.

Str.charCodeAt(pos)- devuelve el código ASCII/UTF-16 de la posición.

Str.split(pm1); - devuelve un array de strings, separado por pm1.

Str.IndexOf(pm1,pm2); - devuelve la posición de donde se encuentra el pm1, pm2 es la localización de donde empezar.

Str.lastIndexOf(pm1); - lo mismo pero empieza la búsqueda al final.

Str.includes(pm1); - devuelve booleano si incluye o no.

Str.startsWith(pm1); - devuelve booleano empieza en pm1.

Str.endsWith(pm1); - devuelve booleano si acaba en pm1.

***Date*** – Objeto para fechas

Let x = new Date(YYYY,MM,DD); - sin parámetros dá la fecha del sistema.

***Array***

Arr.length(); – Devuelve la longitud de un array.

Arr.toString();

Arr.pop(); – elimina el ultimo elemento de un array, devuelve el elemento eliminado.

Arr.push(); – añade un elemento al final de un array.

Arr.shift(); – elimina el primer elemento de un array, devuelve el elemento eliminado y cambia las posiciones de los otros elementos pos 1 -> pos 0.

Arr.unshift(p1); – añade el p1 en el principio de un array, devuelve la nueva longitud.

Arr1.concat(arr2,arr3,…); – devuelve un nuevo array que combina los arrays especificado.

Arr.delete(pos); – elimina la posición especificado, pero deja un “undefined” en la posición.

Arr.splice(p1,p2,p3,…); - se puede usar para eliminar y añadir elementos. P1 – posición en donde añadir/eliminar, p2 – numero de cantitad de elementos que eliminar, p3 …. – elementos que añadir.

Arr.slice(p1,p2); - Devuelve un nuevo array con los parámetros especificados. P1 – posición en donde empezar , p2- posición en donde terminar pero con el valor de posición p2 excluida.

***Set*** – Array pero no deja que los elementos repiten.

Let x = new set();

x.size() – devuelve la longitud.

x.add(p1); - meter p1 en el set.

x.delete(p1); - elimina p1 en el set.

x.has(p1); - boolean, devuelve si existe p1 en el set.

***Map*** - key:value array

Let x = new Map();

Let x = new Map([ [key1,val1], [key2,val2], ….. ]);

x.size() – devuelve la longitud.

x.set(k1,v1); - añade un nuevo elemento.

x.get(k); - devuelve el valor de k.

x.delete(k); - elimina k (llave).

x.has(k); - devuelve boolean si existe la llave.

x.entries(); - devuelve un iterador de las llaves y valores.

x.keys(); - devuelve un iterador de las llaves.

x.values(); - devuelve un iterador de los valores.

for (let [key,value] of nombreMap) {} – for of

***Objetos***

Creación de Objetos

let nombreObjeto = new Objeto(); -> Creación de objeto.  
nombreObjeto.nombrePropiedad=valor; -> Asignación de propiedad  
  
Creación de Objetos formato JSON  
let nombreObjeto={  
 propiedad1=””,  
 propiedad2=valor,  
  
 objetoFuncion:function(){código}  
};  
  
***Objeto*** this.propiedad = muestra el valor de la propiedad del objeto.

delete objetoNombre.propiedad; - elimina la propiedad del objeto.  
  
***Classes***

Class nombreClase{

constructor(paraPropiedad1, paraPropiedad2){

this.paraPropiedad1 = paraPropiedad1;

this.paraPropiedad2= paraPropiedad2;

}  
  
 funcionClase(){código}

}

Creación de un objeto de clase

let varClase = new nombreClase(param1,param2);

***Herencia***

Sintaxis de herencia -> claseHijo extends clasePadre

class nombrePadre{

constructor(paramPadre1,paramPadre2){

this.paramPadre1=paramPadre1;

this.patamPadre2=paramPadre2;

}

funcionFamilia(){}

}

class nombreHijo extends nombrePadre{

constructor(paramPadre1,paramHijo1){

super(paramPadre1,paramPadre2);

this.paramHijo1=paramHijo1;

}

funcionFamilia(){ super.funcionFamilia() + codigoHijo;}

}

***Callback*** – Funciones que entran como parámetros en otros funciones. Entra sin paréntesis al final.

setTimeout(cb,milisegundos); -> setTimeout(()=>acción función, 2000); -> Espera el milisegundo antes de que se haga el callback.

arr.sort(cbComparar);  
cbComparar->  
A.Un número negativo si el primer parámetro es menor que el segundo.

B. Cero si son iguales.

C. Un número positivo si el segundo parámetro es mayor que el primero.  
  
arr.forEach(cb(valorActual,Indice,Array)){};  
Ejemplo Sintaxis  
Texto, Carta

Descripción generada automáticamente

forEach Map  
Texto, Carta

Descripción generada automáticamente  
  
arr.map(cb); - devuelve un nuevo array con la condiciones propuesta.  
arr.reduce(cb(acumulador,valorActual,índice[opcional],array[opcional]); - devuelve el mismo array con la función callback ejecutada una vez por cada elemento.  
arr.filter(cb(valor,índice[opcional],array[opcional]); - devuelve un nuevo array cumpliendo la condición del callback (callback debe ser booleano).

***BOM***

Scope  
window – Objeto que representa la ventana de un documento DOM.

***Temporizador***

setTimeout(cb,retraso(milisegundos),[parámetro1 cb,parametro2 cb, …]); - [opcional] Ejecuta el callback tras pasar el milisegundo puesto.  
var idTemporizador = scope.setTimeout(funcion[, retraso, parametro1, parametro2, ...]);  
window.clearTimeout(idTimeout) – elimina/borra el seTimeout hecho.  
  
setInterval(cb,intervaloTiempo(milisegundos),[pm1,pm2]); - ejecuta el callback cada intervalo de tiempo.  
window.clearInterval(idIntervalo) – elimina/birra el setInterval hecho.

***Expresiones* *Regulares* (*REGEX*)**

let pattern = /pattern/letraModifier; || let pattern= /^[a-z][0-9]{1,3}/g;

let pattern = new RegExp(“pattern”,”letra modifier”);

***Funciones******Regex***

RegExp.exec(String); -> Devuelve un array de palabras de match del regex, o un null.

RegExp.test(String);-> Devuelve true o false si contiene un match.

RegExp.toString(RegExp) -> Devuelve el regex como String.

Text.match(RegExp) -> Busca un match en el texto que coincide como match del regex si hay más que una, devuelve un array.

***Modifiers***

g -> Global, buscar todo los matches, no solo el primer match.

i -> Case-Sensitive, Acepta ambos mayúsculas y minúsculas como match.

d-> Especifica en que termina los matches. RegExp.match(/aabb/d);

m-> Busca matches en todas las líneas, no solo en la primera (si hay).

***Grupos***

[] -> Brackets, buscar múltiples matches con las condiciones agrupados.

[abc] -> Busca abc en un texto.

[a-z 0-9] -> Busca el rango de caracteres especificado a – z , 0 - 9 en un texto.

[^a-c] -> No incluye a – c en los matches.

(a|b) -> busca a o b.

***MetaCharacteres***

. -> hace match con cualquier carácter excepto por new line, terminador de línea.

\w -> Match con cualquier carácter de palabra A-Z,a-z,0-9, y \_

\W -> Match con cualquier no carácter de palabra Espacio,símbolos.  
\d -> dígitos, números.

\D -> no dígitos, letras y símbolos.

\s -> WhiteSpace (Espacio, tabulador).

\S -> No whitescape

\b -> Para buscar matches en donde queremos que empieze /termine. /bLO (busca que empieza por LO), LO/b (busca que termine con LO).

\B -> Que no empieza/termine.  
\0 -> Nulo

\n-> Nueva line

\t -> tabulador

***Cuantificadores***

+ -> Una o más.

\*-> 0 o más.

? -> 0 o 1.

{2} -> 2 caracteres/dígitos.

{2,3} -> 2 o 3 caracteres/dígitos.

{2,} -> 2 o más caracteres/dígitos. /\d{2,}/ -> Contiene 2 o más números.

$ -> Termina con el pattern al lado /in$/-> termina con in.

^ -> Empieza con el pattern al lado /^is/ -> Empieza con is

?= -> Hace match con cualquier palabra que tiene el pattern al lado. /is(?=aw)/ -> isaw, isawold, isawi.

?! -> Hace match con cualquier palabra que no tiene el pattern al lado.

***Gestion de Eventos***

Eventos -> Avisos que genera el ordenador en respuesta a acciones de usuarios, o del propio sistema.

*Fases de comunicación de evento*.

Fase de Captura (capturing) -> Recorrido de document hasta el padre del elemento target.

Fase de destino (target) -> Comunicación del evento al elemento target.

Fase de Propagación -> Recorrido de regreso desde el target hasta el document.

*Métodos*

document.elemento.addEventListener(“evento”,método,true/false);

true -> pendiente en fases de captura.

false -> solo atento al destino.

hasFocus() -> Devuelve true/false si el elemento tiene foco.

document.activeElement -> el elemento que tiene foco.

***JSON***

Methods

JSON.parse(objeto); -> Traduce JSON como objeto. Si quieres desde el storage al navegador.

JSON.stringify(); -> JSON a String. Si quieres mandar un objeto desde el storage al navegador.

***Cookie***

Iniciar cookie

document.cookie=”key=value;expires=date”;

Eliminar un cookie

document.cookie=”key=value;expires=date.antiguo”;

***Storage***

***LocalStorage***

localStorage.length()

localStorage.setItem(key,value);

localStorage.getItem(key);

localStorage.key(index); -> devuelve la llave del index.

localStorage.removeItem("key");

localStorage.clear(); -> Elimina el storage.

***Session******Storage***

sessionStorage.length();

sessionStorage.setItem(key,value);

sessionStorage.getItem(key);

sessionStorage.key(index); -> devuelve la llave del index.

sessionStorage.removeItem("key");

sessionStorage.clear();