

## JAVASCRIPT - 1

- Para incluir código JS en una página HTML se usa:

Opción 1:

```
<script LANGUAGE="JavaScript" TYPE="text/javascript">
```

// código JS → Comentarios van igual que en Java y en C

```
<script>
```

pueden ser mayúsculas.

= y /\* \*/

Opción 2:

```
<script>
```

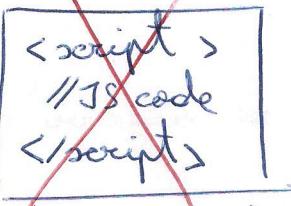
```
</script>
```

- Ejecutar código en un archivo .js

\* El archivo que contiene el código (archivo.js) de tener la forma:

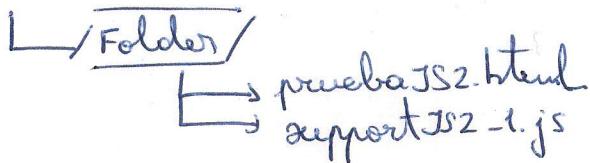


NO debe contener los tags <script> y </script>



\* Referenciar al archivo .js → Se puede incluir tanto en <head> como

en <body>  
Suponiendo que tenemos un archivo supportJS2-1.js, en la  
misma carpeta que en el archivo que se leva a llamar pruebaJS2.html



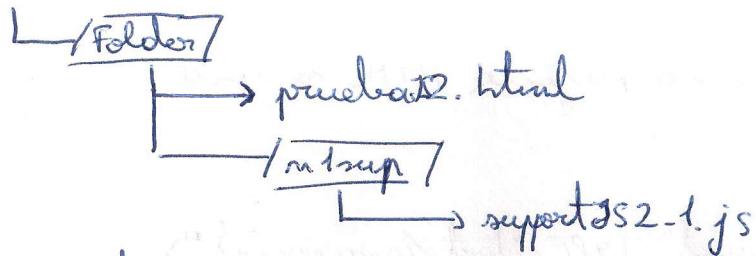
Opción 1:

```
<script LANGUAGE="JavaScript" TYPE="text/javascript">  
SRC="supportJS2-1.js"></script>
```

Opción 2:

```
<script SRC="supportJS2-1.js"></script>
```

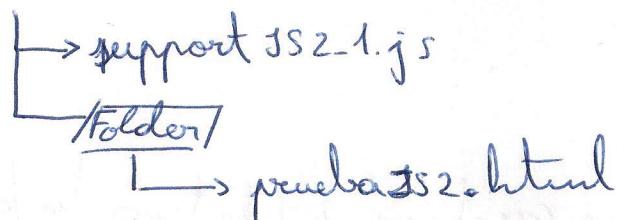
Si la ubicación de los archivos es:



```
<script src="m1sup/supportJS2-1.js"></script>
```

(también se puede usar el formato de tag con más opciones).

Si la ubicación de los archivos es:



```
<script src="../supportJS2-1.js"></script>
```

## COMENTARIOS

//

/\* \*/

NO incluir en los comentarios ~~últimos~~ caracteres cuyo valor supera el valor ASCII 128, como Ñ, ©

## LITERALES

Pueden tomar como valor una variable o una constante, a parte de los distintos tipos de números y valores booleanos.

"una frase"

= 3434

3.4

true, false

Se pueden especificar vectores:

```
vacaciones = ["Navidad", "Semana Santa", "Verano"];
```

```
alert(vacaciones[0]);
```

Lo mostraría Navidad.

## JAVASCRIPT - 2

- Tipos de datos: valor que puede tomar un identificador (una variable o una constante). Si el tipo de dato es fecha, el identificador que tenga ese tipo sólo podrá almacenar fechas.  
En JS los tipos de datos se asignan dinámicamente.

### - Variables:

- Deben comenzar por una letra o un subrayado (-), pudiendo haber más dígitos entre los demás caracteres.
- Se declara por medio de la palabra 'var'.
- Una variable cuando NO es declarada tiene siempre ámbito global.

```
var x; //Accesible fuera y dentro de pruebas.
```

```
y = 2; //Accesible fuera y dentro de pruebas.
```

```
function pruebas () {
```

```
    var = z; //Accesible sólo dentro de pruebas.
```

```
    w = 1; //Accesible fuera y dentro de pruebas.
```

```
}
```

- Pedirar varias variables en una única sentencia:

```
var x, y, z;
```

### - REFERENCIAS

Apuntan a otras variables.

- Ejemplo de referencias con funciones:

```
function fIE7 () { ... }
```

```
function fIE8 () { ... }
```

```
var funcion = (IE7) ? fIE7 : fIE8;
```

```
funcion();
```

Lo si la función/método tiene parámetros sería:

```
función (parámetros);
```

## - Parámetros a funciones:

función `f1(param)`

→ solo accesible dentro de la función.

`param++;`

}

`f1("99");`

`f1(99);` → en ambas llamadas se incrementa su valor.

`f1("xxx")` → provoca que después de `param++` se obtenga `Nan`.

`f()` → se obtiene "undefined"

utilizar: `function f()`

`param = param + 2;`

`window.alert(param + " --- ");`

provoca que no se vea nada por pantalla.

## - VECTORES Y MATRICES → van del 0 al N-1

Son objetos ⇒ se declaran utilizando "new".

\* Formas de declarar un vector en JavaScript:

1) `var v1 = [];` ó `var v1 = [1, 2, 3, 4];`

2) `var v2 = new Array(x);` x puede ser un número directamente  
o bien puede ser una variable declarada como `var x=4;`

\* Si se declara:

`var v2 = new Array(5);`

y se necesita ampliar el array, ¿Qué hay que hacer:

`v2[5] = 9;`

`v2[6] = 10;` → en este caso, en el momento de listar los

`v2[8] = 8;` dentro, `v[7]` no aparecerá porque no tiene ningún  
valor.

## JAVASCRIPT - 3

### - VECTORES Y MATRICES:

\* Listar los elementos: for-each

```
var a = [];
a[0] = 1;
a[2] = 3;
a[5] = 5;
a[4] = 4;

var key;
for (key in a) {
    window.alert(key + " --- " + a[key]);
}
```

Listará todos los elementos que tengan valor y en el orden que ocupan en el vector (no en el orden de asignación como ocurre en PHP).

\* Identificadores ó índices:

Pueden ser índices ó palabras:

```
var a = [];
a["lunes"] = 1;
a["martes"] = 2;
a["jueves"] = 4;           → en este caso la posición "jueves" va
a["miércoles"] = 3;       antes que la posición "miércoles".
```

### - Operadores

- Menos unario: Símbolo: - Uso: -(2+4) Resultado: -6

- De incremento:

a = 1; → a vale 2  
b = ++a;  
-----

a = 1; → a vale 2, b vale 1  
b = a++;

## - Operadores de comparación:

Igualdad:  $2 == '2'$  → True

Igualdad estricta:  $2 === '2'$  → False / Desigualdad estricta  
(comprueba el tipo)

## - Operadores de asignación

$x += y;$  →  $x = x + y;$

$x -= y;$  →  $x = x - y;$

$x *= y;$  →  $x = x * y;$

$x /= y;$  →  $x = x / y;$

$x \% y;$  →  $x = x \% y;$

## - OBJETOS

`new` → crea un objeto.

`delete` → borra un objeto.

`this` **■**

Para tratar sus propiedades en un bucle, en lugar de usar `objeto.propiedad` diverse veces se utiliza la estructura `with`:

```
with (objeto){  
    propiedad1 = ...  
    propiedad2 = ...  
    ...  
}
```

- Funciones → se le pasan los parámetros por valor ⇒ si se modifica el valor de alguno de los parámetros dentro de la función, este cambio no se verá fuera.

## JAVASCRIPT - 4

- Var args en JavaScript:

```
function nombreFuncion(param1, param2, ..., paramN) {  
    for(i=0; i<arguments.length; i++)  
    }  
    ↑  
    para controlar el número de argumentos.
```

- NO EXISTE EL CONCEPTO DE INSTANCIA Y OBJETO, ÚNICAMENTE EL DE OBJETO.

- OBJETOS

Ejemplo de como crear una clase:

```
function NombreObjeto (param1, param2){  
    this.param1 = param1;  
    this.param2 = param2;
```

- ARRAY → Un array puede contener diferentes tipos de objetos.

\* Declaraciones:

vector = new Array (longitud); → crear un vector con un número especificado de elementos.

vector = new Array (elem1, elem2, ..., elemN); → vector que contiene los elem. indicados y longitud N.

\* Propiedades:

- length: contiene el número de elementos del vector.

- concat(vector2): añade los elementos de vector2 al final de los del vector que invoca el método, devolviendo el resultado.

~~sort~~ - sort (funciónComparación): ordena los elementos del vector alfabéticamente. Si se añade una función de comparación como parámetro los ordenará utilizando ésta. Dicha función debe aceptar dos parámetros y:

- devolver  $\varnothing$  si son iguales.
- devolver  $<\varnothing$  si el primer parámetro es menor que el segundo.
- devolver  $>\varnothing$  si el primer parámetro es mayor que el segundo.

```
function compararEnteros(a,b){
```

```
    return a < b ? -1 : (a == b ? 0 : 1); }
```

Usando esta función ordenaría numéricamente (y de menor a mayor) los elementos del vector.

## - Objeto String:

Forma de declararlo: cadena = "semana, frase, persona";

\* ~~objeto~~ string.indexOf(subcadena)

Daruelve la pés empezando a contar en \$.

card = "a b c d e f g h i";  
      |  
      |

cad. ends ~~of~~ ("efg"); → derivative 4

## - Objets Function:

```
function = new Function ([arg1,arg2,...,argN], codigos);
```

Ejemplo:

```
f1 = new Function ([cadena], "return cadena == 'Daniel' ? true : false");
```

## - EVENTOS

Estos controladores se asocian a un elemento HTML y se incluyen así:

[textos](http://www.allianz.es)

función a ejecutar cuando se pasa el mouse por encima del texto. La función `Hifuncion` está en la zona de código JS.

Definición mediante código:

```
<BODY onLoad="Saludos()">
```

</BODY>

<script>

function Saludo() { ... } → esta función puede estar después  
</script> de <BODY></BODY>

-----  
Este se puede traducir como:

<script>

<script>  
!-- Los comentarios de este tipo ocultan el código a  
navegadores sin JavaScript 

```
function Saludo () {  
    alert ("Hola");
```

## JAVASCRIPT - 5

```
window.onload = function () {  
    Saludo();  
}  
// -->  
</script>
```

→ puede NO ponerse este comentario en Java.

Tanto si el código se pone en el BODY como en el HEAD, se ejecutará.

### - VARIABLES

- \* Son case sensitive (diferencia entre mayúsculas y minúsculas).
- \* Pueden empezar con una letra.
- \* Pueden empezar por "\$" ~~o #~~ y "-" (pero no se usarán).
- \* Variable de texto → se puede usar ("") o bien ('').

Declaración de una variable → se utiliza la palabra var.

var nombre\_variable; → ahora la variable está vacía (undefined)

\* Se pueden declarar muchas variables en una línea o en varias:

Ej 1: var lastname="Campos", age=29, name="Jesús";

Ej 2: var lastname="Campos",  
 age=29,  
 name="Jesús";

Re-declaración de una variable:

```
var carname="Volvo";  
var carname;
```

la variable carname aún sigue almacenando el valor "Volvo" en su interior.

### - Bucle For-each (For/In):

```
var person = { name: "Jesús", lname: "Campos", age: 29 };
```

```
for (x in person) {
```

|               |                |
|---------------|----------------|
| person [x]    | x              |
| ↓             | ↓              |
| representa a: | represents to: |
| Jesús         | name           |
| Campos        | lname          |
| 29            | age            |

## - Bloques Try/Catch:

```
try {
```

  abcd; → esto provoca un error.

```
} catch (err) {
```

  var txt = err.message;

  alert (txt);

```
}
```

## - Temporización

```
setTimeout ( function () { alert ("Hello") }, 3000);
```

al cabo de 3 segundos saltará un alert.

## - Comillas: si se introducen comillas dobles o simples se ha de tener en cuenta si son las mismas que encierran el string. Su uso debe ser como sigue (ejemplos).

```
var ans = "It's all right";
```

```
var ans = 'He is called "JC"'; o var ans = "
```

```
var ans = "He is called 'JC'";
```

```
var ans = "He is called \"JC\""; } si las coma que se utilizan para la
```

```
var ans = 'He is called \'JC\''; citación son las mismas que crean el
```

string se debe utilizar el carácter de escape.

## - STRINGS: MATCHING - match - case sensitive

```
var str = "Hello world!";
```

str.match ("world"); → devuelve "world"

str.match ("Word"); → devuelve null.

## - STRINGS: FINDING STRING IN A STRING

indexOf() } ambos devuelven la posición de la primera letra de  
lastIndexOf() } la coincidencia (primera posición). Se empieza a contar  
en el 1.

```
var str = "Hello world, welcome";
```

var str.indexOf ("Hello"). → devolverá 1.

Si no encuentra nada devolverá -1.

## JAVASCRIPT - 6

### - STRINGS: REEMPLAZAR

BACKSLASH (\)

SLASH (/)

str = ".....";

var n = str.replace (areemplazar, valor a poner);  
                    strA                  strB

Si encuentra el strA, lo reemplazará por el strB y n valdrá str pero con los cambios.

Si no lo encuentra n = str.

### - STRINGS: STRING TO ARRAY

txt = "a,b,c,d";

txt.split(","); → separador = coma

txt.split(" "); → separador = espacio

txt.split("|"); → separador = pipe

### - STRING : STRING O OBJETO

var x = "John"; → su tipo es un String

var y = new String ("John"); → su tipo es un objeto