Variables

A la hora de dar nombres a las variables, tendremos que poner nombres que realmente describan el contenido de la variable. No podremos usar palabras reservadas, ni símbolos de puntuación en el medio de la variable, ni la variable podrá contener espacios en blanco. Los nombres de las variables han de construirse con caracteres alfanuméricos y el carácter subrayado (_). No podremos utilizar caracteres raros como el signo +, un espacio, % , $, etc. en los nombres de variables, y estos nombres no podrán comenzar con un número.

https://msdn.microsoft.com/es-es/library/67defydd(v=vs.94).aspx

```
var edad;
edad = 38; // Ya que no estamos obligados a declarar la variable pero es una buena práctica el hacerlo.
var altura, peso, edad: // Para declarar más de una variable en la misma línea.
var mi_peso;
var miPeso;  // Esta opción es más recomendable, ya que es más cómoda de escribir.
```

"use strict" Directive. Modo estricto

"use strict"

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

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

Tipos de variables

- **Cadena**. "Hola mundo". Una serie de caracteres dentro de comillas dobles.
- **Número**. 9.45 Un número sin comillas dobles.
- **Boolean**. true. Un valor verdadero o falso.
- **Null** null. Desprovisto de contenido, simplemente es un valor null.
- **Object**. Es un objeto software que se define por sus propiedades y métodos (los arrays también son objetos).
- **Function**. La definición de una función.

var y let

- **let** define una variable local limitando su alcance al bloque de ejecución, expresión o declaración en la que se encuentre. Es una característica no estándar por lo que puede dar problemas en diferentes navegadores.
- **var** define una variable limitando su ámbito a la función en la que se define o al ámbito global (si no se encuentra dentro de una función), independientemente del bloque de ejecución en el que se ejecute.

```
var a = 5;
var b = 10;

if (a === 5) {
  let a = 4; // El alcance es dentro del bloque if
  var b = 1; // El alcance es global

  console.log(a);  // 4
  console.log(b);  // 1
}

console.log(a); // 5
console.log(b); // 1
```

Conversión

Para convertir cadenas a números dispones de las funciones: parseInt() y parseFloat().

Por ejemplo:

```
parseInt("34")          // resultado = 34
parseInt("89.76")          // resultado = 89
```

parseFloat devolverá un entero o un número real según el caso:

```
parseFloat("34")          // resultado = 34
parseFloat("89.76")     // resultado = 89.76

4 + 5 + parseInt("6")        // resultado = 15
```

Si lo que deseas es realizar la conversión de números a cadenas, es mucho más sencillo, ya que simplemente tendrás que concatenar una cadena vacía al principio, y de esta forma el número será convertido a su cadena equivalente:

```
(" " + 3400)          // resultado = "3400"
(" " + 3400).length     // resultado = 4
```

Categorías de operadores en JavaScript

Comparación.

Comparan los valores de 2 operandos, devolviendo un resultado de true o false (se usan extensivamente en sentencias condicionales como if... else y en instrucciones loop).

```
==  !=  === !==  >  >=  <  <=
```

Aritméticos.

Unen dos operandos para producir un único valor que es el resultado de una operación aritmética u otra operación sobre ambos operandos.

```
+   -   *   /   %   ++   - -   +valor  -valor
```

Asignación.

Asigna el valor a la derecha de la expresión a la variable que está a la izquierda.

```
=   +=   -=   *=   /=   %=   <<=   >=   >>=   >>>=   &=   |=   ^ =
[]
```

Boolean.

Realizan operaciones booleanas aritméticas sobre uno o dos operandos booleanos.

```
&&  ||  !
```

Bit a Bit.

Realizan operaciones aritméticas o de desplazamiento de columna en las representaciones binarias de dos operandos.

```
&   |   ^ ~ <<   >>   >>>
```

Objeto.

Ayudan a los scripts a evaluar la herencia y capacidades de un objeto particular antes de que tengamos que invocar al objeto y sus propiedades o métodos.

```
.   []   ()   delete   in   instanceOf   new   this
```

```
var  s  =  new  String('rafa');
var  longitud  =  s.length;
var  pos  =  s.indexOf("fa");       //  resultado:  pos  =  2
```

- **corchetes** para enumerar miembros de un objeto

```
var a =["Santiago","Coruña", "Lugo"]; // Por ejemplo cuando creamos un array:
a[1] = "Coruña"; // Enumerar un elemento de un array:
a["color"]= "azul"; // Enumerar una propiedad de un objeto:
```

Delete (para eliminar un elemento de una colección).

Por ejemplo si consideramos:

```
var oceanos = new Array("Atlantico", "Pacifico", "Indico","Artico");
```

Podríamos hacer:

```
delete oceanos[2];
// Esto eliminaría el tercer elemento del array ("Indico"), pero la longitud del
array no cambiaría. <br />// Si intentamos referenciar esa posición oceanos[2]
obtendríamos undefined.
```

In (para inspeccionar métodos o propiedades de un objeto).

El operando a la izquierda del operador, es una cadena referente a la propiedad o método (simplemente el nombre del método sin paréntesis); el operando a la derecha del operador, es el objeto que estamos inspeccionando. Si el objeto conoce la propiedad o método, la expresión devolverá true.

```
"write" in document
"defaultView" in document
```

instanceof (para comprobar si un objeto es una instancia de un objeto nativo de JavaScript).

```
a = new Array(1,2,3);

a instanceof Array; // devolverá true.
```

new (para acceder a los constructores de objetos incorporados en el núcleo de JavaScript).

```
var hoy = new Date();

// creará el objeto hoy de tipo Date() empleando el constructor por defecto de
dicho objeto.
```

this (para hacer referencia al propio objeto en el que estamos localizados).

```
<button id="nombre" onclick="document.write(this.value);" value="Adios"><br> Hola
y ...</button><br><br><br />// <code>Tenemos un botón que al pulsarlo escribe
adiós. Escribe this.value, que en el objeto actual que es botón
// con id nombre. Es decir el value del objeto actual.
```

Misceláneos.

Operadores que tienen un comportamiento especial.

El operador coma ,

```
var nombre, direccion, apellidos, edad;

for  (var  i=0,  j=0 ;  i  <  125;  i++,  j+10)
{
     // más instrucciones aquí dentro
}
```

? : (operador condicional)

Este operador condicional es la forma reducida de la expresión if .... else. La sintaxis formal para este operador condicional es:

condicion ? expresión si se cumple la condición: expresión si no se cumple;

Si usamos esta expresión con un operador de asignación:

var = condicion ? expresión si se cumple la condición: expresión si no se cumple;

Ejemplo:

```
     var a,b;
     a = 3; b = 5;
     var h = a > b ? a : b;      // a h se le asignará el valor 5;
```

typeof (devuelve el tipo de valor de una variable o expresión).

Este operador unario se usa para identificar cuando una variable o expresión es de alguno de los siguientes tipos: number, string, boolean, object, function o undefined.

Ejemplo:

```
if  (typeof  miVariable  ==  "number")
{
     miVariable  =  parseInt(miVariable);
}
```

Estructuras de control.

Construcción if

```
if (miEdad >30)
{
     alert("Ya eres una persona adulta");
}
```

Construcción if ... else

```
if (miEdad >30)
{
     alert("Ya eres una persona adulta.");
}
else
{
     alert("Eres una persona joven.");
}
```

Construcción switch

switch (variable) { valor1: // Instrucciones que se van a realizar break; // Rompemos para no ejecutar el resto valor2: // Instrucciones que se van a realizar break; ..... // Más comprobaciones valorn: // Instrucciones a realizar break; default: // Que se hacen en le resto de los casos // No hay break ya que es la última. }

break rompe tanto bucles como switch y deja de ejecutar ese bloque. Utilizar break no es una buena práctica, salvo en switch.

```
accion=prompt("¿Qué desea realizar?","Nada"); // Esta acción va a pedir una
variable al usuario a través de una
```

```
                                                    // ventana similar a alert, pero que
permite introducir datos.

switch (accion) {
     case "Saltar": document.write("Acabas de Saltar"); break;
     case "Gritar": document.write("Es imposible que puedas gritar más fuerte.");
break;
     case "Trotar": document.write("Pareces un caballo de carreras."); break;
     case "Picar": document.write("Le coges a tu compañero una bolsa de patatas");
break;
     default: document.write("Elige algo que puedas hacer la próxima vez");
}
// Se puede comprobar, no es necesario que sean números enteros.<br />
```

Bucles.

for

```
for  (var  i=1;  i<=20;  i++)
{
     // instrucciones que se ejecutarán 20 veces.
}
```

Bucle while().

```
var i=0;
while (i <=10)
{
     // Instrucciones a ejecutar dentro del bucle hasta que i sea mayor que 10 y
no se cumpla la condición.
     i++;
}
```

Bucle do ... while().

```
var a = 1;
do{
     alert("El valor de a es: "+a);     // Mostrará esta alerta 2 veces.
     a++;
}while (a<3);
```

JavaScript HTML DOM Events

DOM

Eventos

```
<body onload="calculaLetra()">
<button type="button" onclick="calulaNota()">Ver nota final</button>
```

Finding HTML Elements. getElementById

```
var x = document.getElementById("nota").value;
document.getElementById("nota_texto").innerHTML = text;
```

JavaScript Popup Boxes

```
var dni_num = prompt("Introduce número de DNI");  // Prompt Box
alert("Incorrecto"); // Alert Box

if (confirm("Press a button!")) { // Confirm Box
    txt = "You pressed OK!";
} else {
    txt = "You pressed Cancel!";
}
```

Debugger

https://www.youtube.com/watch?v=KsF01VJN3RM&feature=youtu.be

Testing javascript

- jet
- mocha

Enlaces

Extensioens de Visual Studio Code

Javascript básico

- Tutorial de HTML5: http://www.w3schools.com/html/html5_intro.asp
- Integración de Javascript en HTML5: http://www.w3schools.com/js/js_whereto.asp
- Sintaxis general: http://www.w3schools.com/js/js_syntax.asp
- Cometarios: http://www.w3schools.com/js/js_comments.asp
- Variables: http://www.w3schools.com/js/js_variables.asp
- Tipos de datos: http://www.w3schools.com/js/js_datatypes.asp

Operadores:

- Operadores en general.http://www.w3schools.com/js/js_operators.asp
- Operadores artiméticos.http://www.w3schools.com/js/js_arithmetic.asp
- Operadores de Asignación.http://www.w3schools.com/js/js_assignment.asp
- Operadores booleanos.http://www.w3schools.com/js/js_booleans.asp
- Operadores de comparación.http://www.w3schools.com/js/js_comparisons.asp
- Condicional "if": http://www.w3schools.com/js/js_if_else.asp
- Condicional "switch": http://www.w3schools.com/js/js_switch.asp
- Bucle "for": http://www.w3schools.com/js/js_loop_for.asp
- Bucles "while" y "do-while": http://www.w3schools.com/js/js_loop_while.asp
- Break y continue no son recomendables para una buena programación, pero para entenderlos mire aquí: http://www.w3schools.com/js/js_break.asp
- Depurar códifo con DevTools de Chrome https://developers.google.com/web/tools/chrome-devtools/javascript/?hl=es
- Framework de realización de pruebas unitarias, para facilitar la ejecución y comprobación de código JavaScript: https://qunitjs.com/
- Documentación JavaScripts JSDoc http://usejsdoc.org/about-getting-started.html