**BUCLES E ITERACIÓN**

Usted puede pensar en un bucle, como en una versión de ordenador del juego en el que se le dice a alguien que dé X pasos en una dirección, y que dé Y pasos en otra; por ejemplo, la idea "Vaya cinco pasos al Este" podría ser expresada como un bucle de esta forma:

var paso;

for (paso = 0; paso < 5; paso++) {

// Se ejecuta 5 veces, con valores desde paso desde 0 hasta 4.

console.log('Dando un paso al Este');

};

Hay muchas clases diferentes de bucles, pero todos ellos hacen en esencia lo mismo: repetir una acción un número de veces (y de hecho es posible que este número sea cero). Los diferentes bucles ofrecen también diferentes formas de determinar sus puntos de inicio y final. Hay varias situaciones que son más fácilmente resueltas por un tipo de bucle que por otros.

Las sentencias para bucles disponibles en JavaScript son:

* [sentencia for](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Bucles_e_iteraci%C3%B3n#sentencia_for)
* [sentencia do...while](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Bucles_e_iteraci%C3%B3n#sentencia do...while)
* [sentencia while](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Bucles_e_iteraci%C3%B3n#sentencia while)
* [sentencia label](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Bucles_e_iteraci%C3%B3n#sentencia label)
* [sentencia break](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Bucles_e_iteraci%C3%B3n#sentencia break)
* [sentencia continue](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Bucles_e_iteraci%C3%B3n#sentencia continue)
* [sentencia for...in](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Bucles_e_iteraci%C3%B3n#sentencia for...in)
* [sentencia for...of](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Bucles_e_iteraci%C3%B3n#sentencia for...of)

**SENTENCIA for**

Un [bucle](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for) [for](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for)se repite hasta que la condición especificada se evalúa como false. El bucle for en JavaScript es similar al de Java y C. Una sentencia for se muestra como sigue:

for ([expresionInicial]; [condición]; [expresionIncremento])

sentencia

Cuando un bucle for se ejecuta, ocurre lo siguiente:

1. La expresión de inicialización expresionInicial, si existe, se ejecuta. Esta expresión habitualmente inicializa uno o más contadores del bucle, pero la sintaxis permite una expresión con cualquier grado de complejidad. Esta expresión puede también declarar variables.
2. Se evalúa la expresión condición. Si el valor de condición es true, se ejecuta la sentencia del bucle. Si el valor de condición es false, el bucle for finaliza. Si la expresión condición es omitida, la condición es asumida como verdadera.
3. Se ejecuta la sentencia. Para ejecutar múltiples sentencias, use un bloque de sentencias ({ ... }) para agruparlas.
4. Se ejecuta la expresión expresionIncremento, si hay una, y el control vuelve al paso 2.

### **Ejemplo**

La siguiente función contiene una sentencia for que cuenta el número de opciones seleccionadas en una lista (un elemento [<select>](https://developer.mozilla.org/es/docs/Web/HTML/Elemento/select) que permite selección múltiple). La sentencia for declara la variable i y la inicializa a cero. Comprueba que i es menor que el número de opciones en el elemento <select>, ejecuta la sentencia siguiente if, e incrementa i en uno tras cada paso por el bucle.

<form name="selectForm">

<p>

<label for="musicTypes">Choose some music types, then click the button below:</label>

<select id="musicTypes" name="musicTypes" multiple="multiple">

<option selected="selected">R&B</option>

<option>Jazz</option>

<option>Blues</option>

<option>New Age</option>

<option>Classical</option>

<option>Opera</option>

</select>

</p>

<p><input id="btn" type="button" value="How many are selected?" /></p>

</form>

<script>

function howMany(selectObject) {

var numberSelected = 0;

for (var i = 0; i < selectObject.options.length; i++) {

if (selectObject.options[i].selected) {

numberSelected++;

}

}

return numberSelected;

}

var btn = document.getElementById("btn");

btn.addEventListener("click", function(){

alert('Number of options selected: ' + howMany(document.selectForm.musicTypes))

});

</script>

**SENTENCIA while**

La sentencia [do...while](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/do...while) se repite hasta que una condición especificada se evalúa a false. Una sentencia do...while se mostrará como sigue:

do

sentencia

while (condición);

sentencia se ejecuta antes de que la condición sea evaluada. Para ejecutar múltiples sentencias, use un bloque de sentencias ({ ... }) para agruparlas. Si condición es true, la sentencia se ejecuta de nuevo. Al final de cada ejecución, la condición es comprobada. Cuando la condición es falsa, la ejecución se detiene y el control pasa a la sentencia siguiente al do...while.

### **Ejemplo**

En el siguiente ejemplo, el bucle do itera al menos una vez y vuelve a hacerlo mientras i sea menor que 5.

do {

i += 1;

console.log(i);

} while (i < 5);

**SENTENCIA while**

Una sentencia [while](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/while) ejecuta sus sentencias mientras la condición sea evaluada como verdadera. Una sentencia while tiene el siguiente aspecto:

while (condición)

sentencia

Si la condición cambia a falsa, la sentencia dentro del bucle deja de ejecutarse y el control pasa a la sentencia inmediatamente después del bucle.

La condición se evalúa antes de que la sentencia contenida en el bucle sea ejecutada. Si la condición devuelve verdadero, la sentencia se ejecuta y la condición se comprueba de nuevo. Si la condición es evaluada como falsa, se detiene la ejecución y el control pasa a la sentencia siguiente al while.

Para ejecutar múltiples sentencias, use un bloque de sentencias ({ ... }) para agruparlas.

### **Ejemplo 1**

El siguiente bucle while itera mientras n sea menor que tres:

n = 0;

x = 0;

while (n < 3) {

n++;

x += n;

}

Con cada iteración, el bucle incrementa n y añade ese valor a x. Por consiguiente, x y n toman los siguientes valores:

* Después del primer paso: n = 1 y x = 1
* Después del segundo paso: n = 2 y x = 3
* Después del tercer paso: n = 3 y x = 6

Tras completar el tercer paso, la condición n < 3 ya no es verdadera, por tanto, el bucle termina.

### **Ejemplo 2**

Evite los bucles infinitos. Asegúrese de que la condición en un bucle llegue finalmente a ser falsa; de otra forma, el bucle nunca terminará. Las sentencias en el siguiente bucle while se ejecutan sin fin, porque la condición nunca llega a ser falsa:

while (true) {

console.log("Hello, world");

}

**SENTENCIA label**

Un [label](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/label) proporciona una sentencia con un identificador que permite referirse a él desde cualquier lugar de su programa. Por ejemplo, usted puede usar un label para identificar un bucle, y usar las sentencias break o continue para indicar si el programa debe interrumpir un bucle o continuar su ejecución.

La sintaxis de la sentencia label es:

label :

sentencia

El valor de label puede ser cualquier identificador JavaScript que no sea una palabra reservada. La sentencia que usted identifique con un label podrá ser cualquier sentencia.

### **Ejemplo**

En este ejemplo, el label markLoop identifica a un bucle while.

markLoop:

while (theMark == true) {

doSomething();

}

**SENTENCIA break**

Use la sentencia [break](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/break) para salir de un bucle, switch, o en conjunto con una sentencia label.

* Cuando use break sin un label, finaliza inmediatamente el código encerrado en while, do-while, for, o switch y transfiere el control a la siguiente sentencia.
* Cuando usted use break con un label, termina la sentencia especificada por label.

La sintaxis de la sentencia break es la siguiente:

1. break;
2. break label;

La primera forma de la sintaxis finaliza con lo encerrado por el bucle o switch; la segunda finaliza lo especificado por la sentencia label.

### **Ejemplo** **1**

El siguiente ejemplo itera a través de los elementos en un array hasta que encuentra que un índice de un elemento cuyo valor es elValor:

for (i = 0; i < a.length; i++) {

if (a[i] == elValor) {

break;

}

}

### **Ejemplo 2:**Breaking a un label

var x = 0;

var z = 0

labelCancelLoops: while (true) {

console.log("Outer loops: " + x);

x += 1;

z = 1;

while (true) {

console.log("Inner loops: " + z);

z += 1;

if (z === 10 && x === 10) {

break labelCancelLoops;

} else if (z === 10) {

break;

}

}

}

**SENTENCIA continue**

La sentencia continue puede usarse para reiniciar una sentencia while, do-while, for o label.

* Cuando use **continue sin un label**, este termina la iteración en curso del código encerrado en una sentencia while, do-while, o for y continúa la ejecución del bucle con la siguiente iteración. A diferencia de la sentencia break, continue no termina completamente la ejecución del bucle:
* En un bucle while, salta atrás hasta la condición.
* En un bucle for, salta a la expresionIncremento.
* Cuando use **continue con una etiqueta**, esto se aplica a las sentencias anidadas identificadas con una etiqueta.

La sintaxis de la sentencia continue es la siguiente:

1. continue;
2. continue *label;*

### **Ejemplo 1**

El siguiente ejemplo muestra un bucle while con una sentencia continue que se ejecuta cuando el valor de i es tres. Así, n toma los valores uno, tres, siete, y doce.

i = 0;

n = 0;

while (i < 5) {

i++;

if (i == 3) {

continue;

}

n += i;

}

### **Ejemplo 2.**

### Una sentencia etiquetada checkiandj contiene una sentencia etiquetada checkj. Si se encuentra continue, el programa termina la iteración en curso de checkj y empieza la siguiente iteración. Cada vez que continue es encontrado, checkj reitera hasta que su condición devuelve false. Y cuando devuelve false, el resto de la sentencia checkiandj es completada, y checkiandj reitera hasta que su condición devuelve false. Cuando esto ocurre el programa continúa en la siguiente sentencia después de checkiandj.

Si continue tenía una etiqueta checkiandj, el programa continuaría al principio de la sentencia checkiandj.

checkiandj:

while (i < 4) {

console.log(i);

i += 1;

checkj:

while (j > 4) {

console.log(j);

j -= 1;

if ((j % 2) == 0) {

continue checkj;

}

console.log(j + " is odd.");

}

console.log("i = " + i);

console.log("j = " + j);

}

**SENTENCIA for…in**

La sentencia [for...in](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for...in) itera una variable especificada sobre todas las propiedades enumerables de un objeto. Para cada propiedad distinta, JavaScript ejecuta las sentencias especificadas. Una sentencia for...in será como sigue:

for (variable en objeto) {

sentencias

}

### **Ejemplo**

La siguiente función toma como su argumento un objeto y el nombre del objeto. Entonces itera sobre todas las propiedades del objeto y devuelve una cadena que lista los nombres de las propiedades y sus nombres.

function volcar\_propiedades(obj, obj\_nombre) {

var resultado = "";

for (var i in obj) {

resultado += obj\_nombre + "." + i + " = " + obj[i] + "<br>";

}

resultado += "<hr>";

return resultado;

}

Para un objeto coche con propiedades marca y modelo, resultado sería:

coche.marca = Ford

coche.modelo = Mustang

### **Arrays**

Aunque puede ser tentador usar esto como una forma de iterar sobre elementos [Array](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Array), la sentencia **for...in**devolverá el nombre de las propiedades que usted ha definido además de los índices numéricos. En consecuencia es mejor usar un bucle [for](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for) tradicional con un índice numérico cuando esté iterando sobre arrays, ya que la sentencia **for...in** itera sobre las propiedades definidas por el usuario además de los elementos del array, si usted modifica el objeto Array, por ejemplo añadiendo propiedades personalizadas o métodos.

**SENTENCIA for…of**

**This is a new technology, part of the ECMAScript 2015 (ES6) standard.**  
This technology's specification has been finalized, but check the [compatibility table](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Bucles_e_iteraci%C3%B3n#Browser_compatibility) for usage and implementation status in various browsers.

La sentencia [for...of](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for...of) crea un bucle iterando sobre [objetos iterables](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/iterable)  (incluyendo [Array](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Array), [Map](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Map), [Set](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Set), [argumentos](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions_and_function_scope/arguments) objetos etc..), invocando una iteración personalizada conectando con sentencias para ser ejecutadas por el valor de cada propiedad distinta.

for (variable de objeto) {

sentencia

}

El siguiente ejemplo muestra la diferencia entre un bucle for…of  y un bucle for…in. Mientras for...in itera sobre nombres de propiedades, for...of itera sobre valores de propiedades:

let arr = [3, 5, 7];

arr.foo = "hello";

for (let i in arr) {

console.log(i); // logs "0", "1", "2", "foo"

}

for (let i of arr) {

console.log(i); // logs "3", "5", "7"

}