# 4.3. Sentencias break y continue

La estructura de control for es muy sencilla de utilizar, pero tiene el inconveniente de que el número de repeticiones que se realizan sólo se pueden controlar mediante las variables definidas en la zona de actualización del bucle.

Las sentencias break y continue permiten manipular el comportamiento normal de los bucles for para detener el bucle o para saltarse algunas repeticiones. Concretamente, la sentencia break permite terminar de forma abrupta un bucle y la sentencia continue permite saltarse algunas repeticiones del bucle.

El siguiente ejemplo muestra el uso de la sentencia break:

var cadena = "En un lugar de la Mancha de cuyo nombre no quiero acordarme...";

var letras = cadena.split("");

var resultado = "";

for(i in letras) {

if(letras[i] == 'a') {

break;

}

else {

resultado += letras[i];

}

}

alert(resultado);

// muestra "En un lug"

Si el programa llega a una instrucción de tipo break;, sale inmediatamente del bucle y continúa ejecutando el resto de instrucciones que se encuentran fuera del bucle for. En el ejemplo anterior, se recorren todas las letras de una cadena de texto y cuando se encuentra con la primera letra "a", se detiene la ejecución del bucle for.

La utilidad de break es terminar la ejecución del bucle cuando una variable toma un determinado valor o cuando se cumple alguna condición.

En ocasiones, lo que se desea es saltarse alguna repetición del bucle cuando se dan algunas condiciones. Siguiendo con el ejemplo anterior, ahora se desea que el texto de salida elimine todas las letras "a" de la cadena de texto original:

var cadena = "En un lugar de la Mancha de cuyo nombre no quiero acordarme...";

var letras = cadena.split("");

var resultado = "";

for(i in letras) {

if(letras[i] == 'a') {

continue;

}

else {

resultado += letras[i];

}

}

alert(resultado);

// muestra "En un lugr de l Mnch de cuyo nombre no quiero cordrme..."

En este caso, cuando se encuentra una letra "a" no se termina el bucle, sino que no se ejecutan las instrucciones de esa repetición y se pasa directamente a la siguiente repetición del bucle for.

La utilidad de continue es que permite utilizar el bucle for para filtrar los resultados en función de algunas condiciones o cuando el valor de alguna variable coincide con un valor determinado.

# 4.4. Otras estructuras de control

Las estructuras de control de flujo que se han visto (if, else, for) y las sentencias que modifican su comportamiento (break, continue) no son suficientes para realizar algunas tareas complejas y otro tipo de repeticiones. Por ese motivo, JavaScript proporciona otras estructuras de control de flujo diferentes y en algunos casos más eficientes.

### 4.4.1. Estructura while

La estructura while permite crear bucles que se ejecutan ninguna o más veces, dependiendo de la condición indicada. Su definición formal es:

while(condicion) {

...

}

El funcionamiento del bucle while se resume en: "mientras se cumpla la condición indicada, repite indefinidamente las instrucciones incluidas dentro del bucle".

Si la condición no se cumple ni siquiera la primera vez, el bucle no se ejecuta. Si la condición se cumple, se ejecutan las instrucciones una vez y se vuelve a comprobar la condición. Si se sigue cumpliendo la condición, se vuelve a ejecutar el bucle y así se continúa hasta que la condición no se cumpla.

Evidentemente, las variables que controlan la condición deben modificarse dentro del propio bucle, ya que de otra forma, la condición se cumpliría siempre y el bucle while se repetiría indefinidamente.

El siguiente ejemplo utiliza el bucle while para sumar todos los números menores o iguales que otro número:

var resultado = 0;

var numero = 100;

var i = 0;

while(i <= numero) {

resultado += i;

i++;

}

alert(resultado);

El programa debe sumar todos los números menores o igual que otro dado. Por ejemplo si el número es 5, se debe calcular: 1 + 2 + 3 + 4 + 5 = 15

Este tipo de condiciones "suma números mientras sean menores o iguales que otro número dado") se resuelven muy fácilmente con los bucles tipo while, aunque también se podían resolver con bucles de tipo for.

En el ejemplo anterior, mientras se cumpla la condición, es decir, mientras que la variable i sea menor o igual que la variable numero, se ejecutan las instrucciones del bucle.

Dentro del bucle se suma el valor de la variable i al resultado total (variable resultado) y se actualiza el valor de la variable i, que es la que controla la condición del bucle. Si no se actualiza el valor de la variable i, la ejecución del bucle continua infinitamente o hasta que el navegador permita al usuario detener el script.

### 4.4.2. Estructura do...while

El bucle de tipo do...while es muy similar al bucle while, salvo que en este caso **siempre** se ejecutan las instrucciones del bucle al menos la primera vez. Su definición formal es:

do {

...

} while(condicion);

De esta forma, como la condición se comprueba después de cada repetición, la primera vez siempre se ejecutan las instrucciones del bucle. Es importante no olvidar que después del while() se debe añadir el carácter ; (al contrario de lo que sucede con el bucle while simple).

Utilizando este bucle se puede calcular fácilmente el factorial de un número:

var resultado = 1;

var numero = 5;

do {

resultado \*= numero; // resultado = resultado \* numero

numero--;

} while(numero > 0);

alert(resultado);

En el código anterior, el resultado se multiplica en cada repetición por el valor de la variable numero. Además, en cada repetición se decrementa el valor de esta variable numero. La condición del bucledo...while es que el valor de numero sea mayor que 0, ya que el factorial de un número multiplica todos los números menores o iguales que él mismo, pero hasta el número 1 (el factorial de 5 por ejemplo es 5 x 4 x 3 x 2 x 1 = 120).

Como en cada repetición se decrementa el valor de la variable numero y la condición es que numerosea mayor que cero, en la repetición en la que numero valga 0, la condición ya no se cumple y el programa se sale del bucle do...while.

### 4.4.3. Estructura switch

La estructura if...else se puede utilizar para realizar comprobaciones múltiples y tomar decisiones complejas. Sin embargo, si todas las condiciones dependen siempre de la misma variable, el código JavaScript resultante es demasiado redundante:

if(numero == 5) {

...

}

else if(numero == 8) {

...

}

else if(numero == 20) {

...

}

else {

...

}

En estos casos, la estructura switch es la más eficiente, ya que está especialmente diseñada para manejar de forma sencilla múltiples condiciones sobre la misma variable. Su definición formal puede parecer compleja, aunque su uso es muy sencillo:

switch(variable) {

case valor\_1:

...

break;

case valor\_2:

...

break;

...

case valor\_n:

...

break;

default:

...

break;

}

El anterior ejemplo realizado con if...else se puede rehacer mediante switch:

switch(numero) {

case 5:

...

break;

case 8:

...

break;

case 20:

...

break;

default:

...

break;

}

La estructura switch se define mediante la palabra reservada switch seguida, entre paréntesis, del nombre de la variable que se va a utilizar en las comparaciones. Como es habitual, las instrucciones que forman parte del switch se encierran entre las llaves { y }.

Dentro del switch se definen todas las comparaciones que se quieren realizar sobre el valor de la variable. Cada comparación se indica mediante la palabra reservada case seguida del valor con el que se realiza la comparación. Si el valor de la variable utilizada por switch coincide con el valor indicado por case, se ejecutan las instrucciones definidas dentro de ese case.

Normalmente, después de las instrucciones de cada case se incluye la sentencia break para terminar la ejecución del switch, aunque no es obligatorio. Las comparaciones se realizan por orden, desde el primer case hasta el último, por lo que es muy importante el orden en el que se definen los case.

¿Qué sucede si ningún valor de la variable del switch coincide con los valores definidos en los case? En este caso, se utiliza el valor default para indicar las instrucciones que se ejecutan en el caso en el que ningún case se cumpla para la variable indicada.

Aunque default es opcional, las estructuras switch suelen incluirlo para definir al menos un valor por defecto para alguna variable o para mostrar algún mensaje por pantalla.

# Unidad 10. Avanzado: JavaScript. Estructuras de control

## **Sentencia IF [ELSE]**

La sentencia If evalúa una condición, expresada entre paréntesis ( ). Si esta se cumple, ejecuta el bloque de instrucciones que tiene entre llaves { }.

**if** (condición) {

bloque de instrucciones;

}

También podemos indicar un bloque de instrucciones, con la palabra else, que se ejecutará si no se cumple la instrucción.

**if** (condición) {

bloque de instrucciones si se cumple;

} **else** {

bloque de instrucciones si nos e cumple;

}

Podemos poner varios else, con otras condiciones entre paréntesis. Se irán evaluando todas las condiciones hasta encontrar la primera cierta. Si una se cumple, se ejecutarán sólo esas instrucciones, aunque hubiesen otras condiciones ciertas. Si no se ha cumplido ninguna, se ejecutará el último else, si no tiene condición.

**if** (condición1) {

bloque de instrucciones si se cumple condición1;

} **else** (condición2) {

bloque de instrucciones si se cumple condición2;

} **else** (condición3) {

bloque de instrucciones si se cumple condición3;

} **else** {

bloque de instrucciones si no se ha cumplido ninguna;

}

Por ejemplo:

**if** (nota >= 0 && nota < 5) {

**return** 'Suspenso';

} **else** (nota >= 5 && nota < 6) {

**return** 'Suficiente';

} **else** (nota >= 6 && nota < 7) {

**return** 'Bien';

} **else** (nota >= 7 && nota < 9) {

**return** 'Notable';

} **else** (nota >= 9 && nota <= 10) {

**return** 'Sobresaliente';

} **else** {

**return** 'La nota no es válida. Tiene que estar entre 0 y 10';

}

Las sentencias If se pueden anidar, siempre que una quede completamente dentro de otra. Por ejemplo:

**if** (condición1) {

**if** (condición2) {

si se cumple condición 1 y condición2;

} **else** {

si se cumple condición 1 pero no condición2;

}

} **else** {

si no se cumple condición1

}

Cuando se emplea la sentencia If para asignar un valor a una variable, se suele utilizar la siguiente estructura, más compacta:

variable = (condición) ? valor\_si\_se\_cumple : valor\_si\_no;

Por ejemplo:

calificacion = (nota < 5) ? 'Suspenso' : 'Aprobado';

## **La sentencia SWITCH**

De forma parecida al If se comporta la sentencia Switch. Se le asigna un valor o expresión, y se compara con cada case. Ejecuta el código de la primera coincidencia encontrada. Si no encuentra ninguna, se ejecuta el código de default.

**switch**(expresion) {

**case** valor1:

instrucciones si coincide;

**break**;

**case** valor2:

instrucciones si coincide;

**break**;

**default**:

instrucciones si no coincide ninguna;

}

Por ejemplo:

**function** paginaFlores(categoria){

**switch**(categoria) {

**case** "silvestres":

**return** "silvestres.html";

**break**;

**case** "ornamentales":

**return** "ornamentales.html";

**break**;

**case** "exóticas":

**return** "exoticas.html";

**break**;

**case** "plantas":

**return** "palntas.html";

**break**;;

**default**:

alert('No existe esa categoría');

**return** "index.html";

}

}

A la función anterior, se le pasa como parámetro el nombre de una categoría, y busca en el Switch una coincidencia con el parámetro par adevolver la página de la categoría. Si no encunetra ninguna coincidencia, muestra un mensaje y devuelve la página de inicio.

## **El bucle WHILE**

Los bucles son estructuras fundamentales en programación, ya que permiten repetir una instrucción un número determinado de veces, o hasta que se cumpla una determinada condición (o deje de cumplirse).

En bucle While, evalúa una condición. Si se cumple, ejecuta su bloque de instrucciones, y vuelve a evaluar la condición. Si se cumple, vuelve a ejecutar las instrucciones, y así sucesivamente mientras que la condición se cumpla.

while (condición) {

bloque de sentencias;

}

Por ejemplo, el siguiente bucle mostraría 5 mensajes. La variable ***i*** inicialmente vale 0. En cada vuelta del bucle, se comprueba si es menor de 5. Si lo es, se incrementa en 1 en la primera instrucción (recuerda que ***i++*** equivale a ***i+=1***, que equivale a ***i=i+1***), se muestra el mensaje en la segunda, y vuelve a comenzar el ciclo. Cuando ***i*** sea 5, mostrará el quinto mensaje, y en la siguiente comprobación, la condición será falsa.

**var** i = 0;

**while** (i < 5) {

i++;

alert('Mensaje número ' + i);

}

El bucle While se puede expresar al revés, para que primero ejecute las instruciones, y después evalúe la condición. Esto asegura que por lo menos se ejecuten las instrucciones una vez, y se repitan mientras se cumpla la condición. Su sintaxis es:

**do** {

bloque de sentencias;

} **while** (condición)

## **El bucle FOR**

El bucle FOR se emplea normalmente para contar, para realizar una repetición un número determinado de veces.

Se define con un valor inicial para la variable que actúa como contador, la condición que se ha de cumplir para que el bucle se repita, y la actualización de la variable (normalmente un incremento). Estros tres parámetros, se separan por punto y coma (***;***).

**for** (valor\_inicial; condición; actualización) {

bloque de sentencias;

}

El orden de ejecución sería: asignar el valor inicial, evaluar la condición, si se cumple, ejecutar las sentencias, actualizar la variable, evaluar la condción, si se cumple ejecutar las sentencias, actualizar la variable, etc...

El mismo ejemplo que hemos utilizado con el bucle While para mostrar 5 mensajes, lo podemos expresar con un For como:

**for** (i=1; i<=5; i++) {

alert('Mensaje número ' + i);

}

Una variante del For es la estructura For .. In, que nos permite reccorrer los elementos de un array o un objeto.

**for** (variable **in** objeto) {

bloque de sentencias;

}

Si lo utilizamos con un array, lo que hará el bucle, es que para cada vuelta, variable tomará el valor de un elemento del array, comenzando por el primero, y se repetirá una vez para cada uno.

Por ejemplo:

**function** datosAlumno() {

alumno = new **Array**('Juan', 25, 8.5);

datos = 'Datos del alumno:';

**for** (dato **in** alumno) {

datos = datos + ' ' + dato;

}

**return** datos;

}