Otras Estructuras de Control

JavaScript

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. Así, veremos las estructuras:

- **Estructura while**
- **Estructura do..while**
- **Estructura switch**

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*".

Estructura while

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.

Estructura while

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);
```

Estructura while

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.

Estructura while

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**.

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).

Estructura do...while

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);
```

Estructura do…while

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 bucle **do...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 numero sea 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**.

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 {
...}
```

Estructura switch

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:

Estructura switch

```
switch(variable) {
case valor_1:
 ...
break;
case valor_2:
 ...
break;
...
case valor_n:
...
break;
default:
...
break; }
```

Estructura switch

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;
}
```

Estructura switch

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**.

Estructura switch

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.