Condicionales

IF es una estructura de control utilizada para tomar decisiones. Es un condicional que sirve para realizar unas u otras operaciones en función de una expresión y primero se evalúa una expresión, si da resultado positivo se realizan las acciones relacionadas con el caso positivo.

La sintaxis de la estructura IF es la siguiente.

```
if (expresión) {
  //acciones a realizar en caso positivo
  //...
}
```

Opcionalmente se pueden indicar acciones a realizar en caso de que la evaluación de la sentencia devuelva resultados negativos.

```
if (expresión) {
  //acciones a realizar en caso positivo
  //...
} else {
  //acciones a realizar en caso negativo
  //...
}
```

Las llaves engloban las acciones que queremos realizar en caso de que se cumplan o no las expresiones. Estas llaves han de colocarse siempre, excepto en el caso de que sólo haya una instrucción como acciones a realizar, que son opcionales.

Por ejemplo:

```
if (llueve)
  alert("Cae agua");
```

Sería exactamente igual que este código:

```
if (llueve){
  alert("Cae agua");
}
```

O incluso, igual a este otro:

```
if (llueve) alert("Cae agua");
```

Generalmente, cuando utilizamos las llaves, el código queda bastante más claro, porque se puede apreciar en un rápido vistazo qué instrucciones están dependiendo del caso positivo del if.

Los saltos de línea tampoco son necesarios y se han colocado también para que se vea mejor la estructura. Perfectamente podríamos colocar toda la instrucción IF en la misma línea de código, pero eso no ayudará a que las cosas estén claras.

Ejemplo de condicionales IF.

```
if (dia == "lunes")
```

```
  document.write ("Que tengas un feliz comienzo de semana")
```

Si es lunes nos deseará una feliz semana. No hará nada en caso contrario. Como en este ejemplo sólo indicamos una instrucción para el caso positivo, no hará falta utilizar las llaves (aunque sí sería recomendable haberlas puesto).

Ejemplo :

```
if (credito >= precio) {
  document.write("has comprado el artículo " + nuevoArtículo) //enseño compra
  carrito += nuevoArticulo //introduzco el artículo en el carrito de la compra
  credito -= precio //disminuyo el crédito según el precio del artículo
} else {
  document.write("se te ha acabado el crédito") //informo que te falta dinero
  window.location = "carritodelacompra.html" //voy a la página del carrito
}
```

Este ejemplo comprueba si tiene crédito para realizar una supuesta compra. Para ello analizamos si el crédito es mayor o igual que el precio del artículo, si es así informo de la compra, introduzco el artículo en el carrito y resto el precio al crédito acumulado. Si el precio del artículo es superior al dinero disponible informo de la situación y mando al navegador a la página donde se muestra su carrito de la compra.

Expresiones condicionales
La expresión a evaluar se coloca siempre entre paréntesis y está compuesta por variables que se combinan entre si mediante operadores condicionales. Recordamos que los operadores condicionales relacionaban dos variables y devolvían siempre un resultado boleano. Por ejemplo un operador condicional es el operador "es igual" (==), que devuelve true en caso de que los dos operandos sean iguales o false en caso de que sean distintos.

```
if (edad > 18)
  document.write("puedes ver esta página")
```

En este ejemplo utilizamos en operador condicional "es mayor" (>). En este caso, devuelve true si la variable edad es mayor que 18, con lo que se ejecutaría la línea siguiente que nos informa de que se puede ver el contenido para adultos.

Las expresiones condicionales se pueden combinar con las expresiones lógicas para crear expresiones más complejas. Recordamos que las expresiones lógicas son las que tienen como operandos a los boleanos y que devuelvan otro valor boleano.

Son los operadores negación lógica, Y lógico y O lógico.

```
if (bateria < 0.5 && redElectrica == 0)
  document.write("la notebook se va a apagar en segundos")
```

Lo que hacemos es comprobar si la batería de nuestra supuesta notebook es menor que 0.5 (está casi vacía) y también comprobamos si no tiene red eléctrica (desenchufado). Luego, el operador lógico los relaciona con un Y, de modo que si está casi sin batería Y sin red eléctrica, muestro el mensaje que el equipo se va a apagar.

La estructura if es de las más utilizadas en lenguajes de programación, para tomar decisiones en función de la evaluación de una sentencia.

Sentencias IF anidadas

Para hacer estructuras condicionales más complejas podemos anidar sentencias IF, es decir, colocar estructuras IF dentro de otras estructuras IF. Con un solo IF podemos evaluar y realizar una acción u otra según dos posibilidades, pero si tenemos más posibilidades que evaluar podemos anidar IF's y crear el flujo de código necesario para decidir correctamente.

Por ejemplo, para comprobar si un número es mayor menor o igual que otro, tengo que evaluar tres posibilidades distintas. Primero puedo comprobar si los dos números son iguales, si lo son, ya he resuelto el problema, pero si no son iguales todavía tendré que ver cuál de los dos es mayor. Veamos este ejemplo en código Javascript.

```
var numero1=23
var numero2=63
if (numero1 == numero2){
  document.write("Los dos números son iguales")
}else{
  if (numero1 > numero2) {
    document.write("El primer número es mayor que el segundo")
  }else{
    document.write("El primer número es menor que el segundo")
  }
}
```

El flujo del programa primero evalúa si los dos números son iguales, en caso positivo se muestra un mensaje informando de ello, en caso contrario ya sabemos que son distintos, pero aún debemos averiguar cuál de los dos es mayor. Para eso se hace otra comparación para saber si el primero es mayor que el segundo. Si esta comparación da resultado positivo mostramos un mensaje que el primero es mayor que el segundo, en caso contrario indico que el primero es menor que el segundo.

Las llaves son en este caso opcionales, y sólo ejecutan una sentencia para cada caso. Además, los saltos de línea y tabuladores son también opcionales en todo caso y nos sirven sólo para ver el código de una manera más ordenada.

Mantener el código bien estructurado y escrito de una manera comprensible es muy importante.

Operador IF Ternario

Este operador es un claro ejemplo de ahorro de líneas y caracteres al escribir los scripts. .

Un ejemplo de uso del operador IF se puede ver a continuación.

Variable = (condición) ? valor1 : valor2

Este ejemplo no sólo realiza una comparación de valores, además asigna un valor a una variable. Lo que hace es evaluar la condición (colocada entre paréntesis) y si es positiva asigna el valor1 a la variable y en caso contrario le asigna el valor2. Veamos un ejemplo:

momento = (hora_actual < 12) ? "Antes del mediodía" : "Después del mediodía"

Este ejemplo analiza si la hora actual es menor que 12. Si es así, es antes del mediodía, y asigna "Antes del mediodía" a la variable momento. Si la hora es mayor o igual a 12 es después del mediodía, con lo que se asigna el texto "Después del mediodía" a la variable momento.

Estructura SWITCH de Javascript

Las estructuras de control son la manera con la que se puede dominar el flujo de los programas, para hacer cosas distintas en función de los estados de las variables.

Switch se utiliza para tomar decisiones en función de distintos estados de las variables. Esta expresión se utiliza cuando tenemos múltiples posibilidades como resultado de la evaluación de una sentencia.

Su sintaxis es la siguiente.

```
switch (expresión) {
        case valor1:
        Sentencias a ejecutar si la expresión tiene como valor a valor1
        break
        case valor2:
        Sentencias a ejecutar si la expresión tiene como valor a valor2
        break
        case valor3:
        Sentencias a ejecutar si la expresión tiene como valor a valor3
        break
        default:
        Sentencias a ejecutar si el valor no es ninguno de los anteriores
}
```

La expresión se evalúa, si vale valor1 se ejecutan las sentencias relacionadas con ese caso. Si la expresión vale valor2 se ejecutan las instrucciones relacionadas con ese valor y así sucesivamente, por tantas opciones como deseemos. Finalmente, para todos los casos no contemplados anteriormente se ejecuta el caso por defecto.

La palabra break es opcional, pero si no la ponemos a partir de que se encuentre coincidencia con un valor se ejecutarán todas las sentencias relacionadas con este y todas las siguientes. Es decir, si en nuestro esquema anterior no hubiese ningún break y la expresión valiese valor1, se ejecutarían las sentencias relacionadas con valor1 y también las relacionadas con valor2, valor3 y default.

También es opcional la opción default u opción por defecto.

Ejemplo: indicar que día de la semana es. Si el día es 1 (lunes) sacar un mensaje indicándolo, si el día es 2 (martes) debemos sacar un mensaje distinto y así sucesivamente para cada día de la semana, menos en el 6 (sábado) y 7 (domingo) que queremos mostrar el mensaje "es fin de semana". Para días mayores que 7 indicaremos que ese día no existe.

```
switch (dia_de_la_semana) {
        case 1:
        document.write("Es Lunes")
        break
        case 2:
        document.write("Es Martes")
        break
        case 3:
        document.write("Es Miércoles")
        break
```

```
        case 4:
        document.write("Es Jueves")
        break
        case 5:
        document.write("Es viernes")
        break
        case 6:
        case 7:
        document.write("Es fin de semana")
        break
        default:
        document.write("Ese día no existe")
}
```

El ejemplo es relativamente sencillo, solamente puede tener una pequeña dificultad, consistente en interpretar lo que pasa en el caso 6 y 7, que habíamos dicho que teníamos que mostrar el mismo mensaje. En el caso 6 en realidad no indicamos ninguna instrucción, pero como tampoco colocamos un break se ejecutará la sentencia o sentencias del caso siguiente, que corresponden con la sentencia indicada en el caso 7 que es el mensaje que informa que es fin de semana. Si el caso es 7 simplemente se indica que es fin de semana, tal como se pretendía.

Bucle FOR

El bucle FOR se utiliza para repetir una o más instrucciones un determinado número de veces. De entre todos los bucles, el FOR se suele utilizar cuando sabemos seguro el número de veces que queremos que se ejecute. La sintaxis del bucle for se muestra a continuación.

```
for (inicialización; condición; actualización) {
        //sentencias a ejecutar en cada iteración
}
```

El bucle FOR tiene tres partes incluidas entre los paréntesis, que nos sirven para definir cómo deseamos que se realicen las repeticiones. La primera parte es la inicialización, que se ejecuta solamente al comenzar la primera iteración del bucle. En esta parte se suele colocar la variable que utilizaremos para llevar la cuenta de las veces que se ejecuta el bucle.

La segunda parte es la condición, que se evaluará cada vez que comience una iteración del bucle. Contiene una expresión para decidir cuándo se ha de detener el bucle, o mejor dicho, la condición que se debe cumplir para que continúe la ejecución del bucle.

Por último tenemos la actualización, que sirve para indicar los cambios que queramos ejecutar en las variables cada vez que termina la iteración del bucle, antes de comprobar si se debe seguir ejecutando.

Después del for se colocan las sentencias que queremos que se ejecuten en cada iteración, acotadas entre llaves.

Un ejemplo de utilización de este bucle lo podemos ver a continuación, donde se imprimirán los números del 0 al 10.

```
var i
for (i=0;i<=10;i++) {
        document.write(i)
        document.write("<br>")
}
```

En este caso se inicializa la variable i a 0. Como condición para realizar una iteración, se tiene que cumplir que la variable i sea menor o igual que 10. Como actualización se incrementará en 1 la variable i.

Como se puede comprobar, en una sola línea podemos indicar muchas cosas distintas y muy variadas, lo que permite una rápida configuración del bucle y una versatilidad enorme.

Por ejemplo si queremos escribir los números del 1 al 1.000 de dos en dos se escribirá el siguiente bucle.

```
for (i=1;i<=1000;i+=2)
        document.write(i)
```

Si nos fijamos, en cada iteración actualizamos el valor de i incrementándolo en 2 unidades.

No utilizamos las llaves englobando las instrucciones del bucle FOR porque sólo tiene una sentencia y en este caso no es obligado, tal como pasaba con las instrucciones del IF.

Si queremos contar descendentemente del 200 al 10 utilizaríamos este bucle.

```
for (i=200;i>=10;i--)
        document.write(i)
```

En este caso decrementamos en una unidad la variable i en cada iteración, comenzando en el valor 200 y siempre que la variable tenga un valor mayor o igual que 10.

Bucles WHILE y DO WHILE

Bucle WHILE
Estos bucles se utilizan cuando queremos repetir la ejecución de unas sentencias un número indefinido de veces, siempre que se cumpla una condición.

Sólo se indica, como veremos a continuación, la condición que se tiene que cumplir para que se realice una iteración.

```
while (condición){
	//sentencias a ejecutar
}
```

Un ejemplo de código donde se utiliza este bucle se puede ver a continuación.

```
var color = ""
while (color != "rojo"){
	color = prompt("dame un color (escribe rojo para salir)","")
}
```

Este es un ejemplo de lo más sencillo que se puede hacer con un bucle while. Lo que hace es pedir que el usuario introduzca un color y lo hace repetidas veces, mientras que el color introducido no sea rojo. Para ejecutar un bucle como este primero tenemos que inicializar la variable que vamos utilizar en la condición de iteración del bucle. Con la variable inicializada podemos escribir el bucle, que comprobará para ejecutarse que la variable color sea distinto de "rojo". En cada iteración del bucle se pide un nuevo color al usuario para actualizar la variable color y se termina la iteración, con lo que retornamos al principio del bucle, donde tenemos que volver a evaluar si lo que hay en la variable color es "rojo" y así sucesivamente mientras que no se haya introducido como color el texto "rojo".

Bucle DO...WHILE
El bucle do...while es la última de las estructuras para implementar repeticiones de las que dispone en Javascript y es una variación del bucle while visto anteriormente. Se utiliza generalmente cuando no sabemos cuantas veces se habrá de ejecutar el bucle, igual que el bucle WHILE, con la diferencia de que sabemos seguro que el bucle por lo menos se ejecutará una vez.

La sintaxis es la siguiente:

```
do {
	//sentencias del bucle
} while (condición)
```

El bucle se ejecuta siempre una vez y al final se evalúa la condición para decir si se ejecuta otra vez el bucle o se termina su ejecución.

Ejemplo: para un bucle WHILE :

```
var color
do {
	color = prompt("dame un color (escribe rojo para salir)","")
} while (color != "rojo")
```

Este ejemplo funciona exactamente igual que el anterior, excepto que no tuvimos que inicializar la variable color antes de introducirnos en el bucle. Pide un color mientras que el color introducido es distinto que "rojo".

Ejemplo de uso de los bucles while

En este ejemplo vamos a declarar una variable e inicializarla a 0. Luego iremos sumando a esa variable un número aleatorio del 1 al 100 hasta que sumemos 1.000 o más, imprimiendo el valor de la variable suma después de cada operación. Será necesario utilizar el bucle WHILE porque no sabemos exactamente el número de iteraciones que tendremos que realizar (dependerá de los valores aleatorios que se vayan obteniendo).

```
var suma = 0
while (suma < 1000){
        suma += parseInt(Math.random() * 100)
        document.write (suma + "<br>")
}
```

Break y continue

Existen dos instrucciones que se pueden usar en de las distintas estructuras de control y principalmente en los bucles, que te servirán para controlar dos tipos de situaciones. Son las instrucciones break y continue.:

break: Significa detener la ejecución de un bucle y salirse de él.
continue: Sirve para detener la iteración actual y volver al principio del bucle para realizar otra iteración, si corresponde.

Break
Se detiene un bucle utilizando la palabra break. Detener un bucle significa salirse de él y dejarlo todo como está para continuar con el flujo del programa inmediatamente después del bucle.

```
for (i=0;i<10;i++){
	document.write (i)
	escribe = prompt("dime si continuo preguntando...", "si")
	if (escribe == "no")
	break
}
```

Este ejemplo escribe los números del 0 al 9 y en cada iteración del bucle pregunta al usuario si desea continuar. Si el usuario dice cualquier cosa continua, excepto cuando dice "no", situación en la cual se sale del bucle y deja la cuenta por donde se había quedado.

Continue
Sirve para volver al principio del bucle en cualquier momento, sin ejecutar las líneas que haya por debajo de la palabra continue.

```
var i=0
while (i<7){
	incrementar = prompt("La cuenta está en " + i + ", dime si incremento", "si")
	if (incrementar == "no")
	continue
	i++
}
```

Este ejemplo, en condiciones normales contaría hasta desde i=0 hasta i=7, pero cada vez que se ejecuta el bucle pregunta al usuario si desea incrementar la variable o no. Si introduce "no" se ejecuta la sentencia continue, con lo que se vuelve al principio del bucle sin llegar a incrementar en 1 la variable i, ya que se ignorarían las sentencia que hayan por debajo del continue.

Ejemplo de la sentencia break

Un bucle FOR planeado para llegar hasta 1.000 pero que lo vamos a detener con break cuando lleguemos a 333.

```
for (i=0;i<=1000;i++){
	document.write(i + "<br>")
	if (i==333)
	break;
```

Bucles anidados en Javascript

Anidar un bucle consiste en ingresar ese bucle dentro de otro. La anidación de bucles es necesaria para hacer determinados procesamientos un poco más complejos que los que hemos visto en los ejemplos anteriores.

Un bucle anidado tiene una estructura como la que sigue.

```
for (i=0;i<10;i++){
        for (j=0;j<10;j++) {
        document.write(i + "-" + j)
        }
}
```

La ejecución funcionará de la siguiente manera. Para empezar se inicializa el primer bucle, con lo que la variable i valdrá 0 y a continuación se inicializa el segundo bucle, con lo que la variable j valdrá también 0. En cada iteración se imprime el valor de la variable i, un guión ("-") y el valor de la variable j, como las dos variables valen 0, se imprimirá el texto "0-0" en la página web.

Debido al flujo del programa en esquemas de anidación como el que hemos visto, el bucle que está anidado (más hacia dentro) es el que más veces se ejecuta. En este ejemplo, para cada iteración del bucle más externo el bucle anidado se ejecutará por completo una vez, es decir, hará sus 10 iteraciones. En la página web se escribirían estos valores, en la primera iteración del bucle externo y desde el principio:

0-0
0-1
0-2
0-3
0-4
0-5
0-6
0-7
0-8
0-9

Para cada iteración del bucle externo se ejecutarán las 10 iteraciones del bucle interno o anidado. Hemos visto la primera iteración, ahora vamos a ver las siguientes iteraciones del bucle externo. En cada una acumula una unidad en la variable i, con lo que saldrían estos valores.

1-0
1-1
1-2
1-3
1-4
1-5
1-6
1-7
1-8
1-9

Y luego estos:

2-0
2-1
2-2
2-3
2-4
2-5
2-6
2-7
2-8
2-9

Así hasta que se terminen los dos bucles, que sería cuando se alcanzase el valor 9-9.

Veamos otro ejemplo. Se trata de imprimir en la página las todas las tablas de multiplicar. Del 1 al 9, es decir, la tabla del 1, la del 2, del 3...

```
for (i=1;i<10;i++){
	document.write("<br><b>La tabla del " + i + ":</b><br>")
	for (j=1;j<10;j++) {
	document.write(i + " x " + j + ": ")
	document.write(i*j)
	document.write("<br>")
	}
}
```

Con el primer bucle controlamos la tabla actual y con el segundo bucle la desarrollamos. En el primer bucle escribimos una cabecera, en negrita, indicando la tabla que estamos escribiendo, primero la del 1 y luego las demás en orden ascendente hasta el 9. Con el segundo bucle escribo cada uno de los valores de cada tabla.