Condicionales

Cuando hay que tomar una decisión aparecen las estructuras condicionales.

En nuestra vida diaria se nos presentan situaciones donde debemos decidir: ¿Entro al sitio A o al sitio B ?, etc.

Cuando se presenta la elección tenemos la opción de realizar una actividad o no realizarla.
En una estructura CONDICIONAL SIMPLE, si es verdadera cierta condición hay actividades y si es falsa no hay actividades. En el caso verdadero pueden existir varias operaciones, entradas y salidas, incluso otras estructuras condicionales.
Ejemplo: Realizar la carga de una nota de un alumno. Mostrar un mensaje que aprobó si tiene una nota mayor o igual a 2.5:

```
<html>
<head>
</head>
<body>
<script language="javascript">
var nombre;
var nota;
nombre=prompt('Nombre:','');
nota=prompt('Nota:','');
if (nota>=2.5)
{
  document.write(nombre+' esta aprobado con un '+nota);
}
</script>
</body>
</html>
```

La instrucción a usar es `if` La condición debe ir entre paréntesis. Si la condición se verifica, se ejecutan todas las instrucciones que se encuentran encerradas entre las llaves seguidas al `if`

Para construir condiciones en un `if` podemos utilizar alguno de los siguientes operadores de comparación o de combinación

Comparación

Se pueden comparar variables de distinto tipo

== Devuelve verdadero si son iguales.

!= Devuelve verdadero si son distintos.

=== Devuelve verdadero si son iguales y del mismo tipo.

!== Devuelve verdadero si son distintos o de distinto tipo.

> Devuelve verdadero si la variable de la izquierda es mayor que la variable de la derecha

< Devuelve verdadero si la variable de la derecha es mayor que la variable de la izquierda

>= Devuelve verdadero si la variable de la izquierda es mayor o igual que la variable de la derecha

<= Devuelve verdadero si la variable de la izquierda es menor o igual que la variable de la derecha

Combinación

&& significa Y

|| significa O

! significa NO

Cuando ya es obligado el resultado, no sigue mirando:

verdadero || (lo que sea) da verdadero.

falso && (lo que sea) da falso.

Condicional compuesto

Cuando se presenta la elección tenemos la opción de realizar una actividad u otra. Es decir tenemos actividades en el caso verdadero y en el falso. NUNCA se realizan las actividades de las dos ramas. En una estructura condicional compuesta tenemos entradas, salidas, operaciones, tanto en el caso del verdadero como en el del falso.

Ejemplo: Realizar un programa que lea dos números distintos y muestre el mayor de ellos:

```
<html>
<head>
</head>
<body>
<script language="javascript">
var num1,num2;
num1=prompt('Primer número:','');
num2=prompt('Segundo número:','');
num1=parseInt(num1);
num2=parseInt(num2);
if (num1>num2)
{
  document.write('el mayor es '+num1);
}
else
{
  document.write('el mayor es '+num2);
}
</script>
</body>
</html>
```

La función prompt retorna un texto por lo que debemos convertirlo a entero cuando queremos saber cual de los dos valores es mayor numéricamente. Recuerda que en el lenguaje Javascript una variable puede ir cambiando el tipo de dato que almacena a lo largo de la ejecución del programa.
Estamos en presencia de una ESTRUCTURA CONDICIONAL COMPUESTA ya que tenemos actividades en el caso verdadero y en el falso.

La estructura condicional compuesta sigue el siguiente esquema:

```
if (<condición>)
{
  <Instruccion(es)>
}
else
{
  <Instruccion(es)>
}
```

Es igual que la estructura condicional simple salvo que aparece la palabra clave “else” y posteriormente un bloque { } con una o varias instrucciones.
Si la condición del if es verdadera se ejecuta el bloque que aparece después de la condición; en caso de que la condición resulte falsa se ejecuta la instrucción o bloque de instrucciones que indicamos después del else.

Otro ejemplo:

```
if (x == 3) {
alert ( "X es igual a 3 )
}
else {
alert ( "X no es igual a 3")
}
```

Observemos que el primer mensaje de alerta se dará solo si la variable x es igual a 3, de otra forma el mensaje de alerta indicará que x no es igual a 3.

Si quieres añadir una condición al elemento else, entonces escribe else if:

```
if (x ==3){
alert("X igual a 3")
}
else if (x==4) {
alert ("X igual a 4")
}
else {
alert ("X no es 3 ni 4")
}
```

En este ejemplo, si x no es igual a 3 ni tampoco a 4 entonces se ejecutará el último mensaje de x no es 3 ni 4. En las comparaciones hay que utilizar el doble signo de igual (= =).

También para casos compuestos está la instrucción "switch" .

La instrucción "switch" toma una variable, y la compara con unos posibles valores:

```
switch(variable) {
case valor1:
acciones1;
break;
case valor2:
acciones2;
break;
.........
.........
case valorN:
accionesN;
break;
default acciones;
}
```

Veamos cada una de las partes:

case valor1: en el caso de que la variable tenga el valor "valor1", realizará las acciones "acciones1".

break: si no se incluye esta instrucción después de cada "case", se realizarían todos los cases del "switch" hasta el final. Con ella sólo se realizarán las acciones referentes al "case" concreto.

default: si el valor de la variable no concuerda con ningún case, se realizarán las acciones de default.

Ciclos

Existe un tipo de estructuras muy importantes, que son las estructuras REPETITIVAS.
Una estructura repetitiva permite ejecutar una instrucción o un conjunto de instrucciones varias veces.

Una ejecución repetitiva de instrucciones se caracteriza por:

- La o las instrucciones que se repiten.
- El test o prueba de condición antes de cada repetición, que hará que se repitan o no las

instrucciones.

Hay tres instrucciones para ciclos: for, while y do-while

For

`for` sirve para establecer un ciclo manejado por una variable, con tres partes principales:

```
for (<Inicialización> ; <Condición> ; <Paso a siguiente vuelta>)
{
  <Instrucciones>
}
```

Las tres partes son: inicialización, condición y paso a siguiente vuelta.

Funcionamiento:

1. Primero se ejecuta una única vez la inicialización. Muchas veces se inicializa una variable.
2. El segundo paso es comprobar la (Condición); en caso de ser verdadera se ejecuta el bloque, en caso contrario continúa el programa detrás de las llaves del for.
3. El tercer paso es la ejecución de las instrucciones del bloque.
4. El cuarto paso es ejecutar el tercer campo (Paso a siguiente vuelta).
5. Luego se repiten sucesivamente del Segundo al Cuarto Paso.

Importante: no lleva punto y coma al final del paréntesis del for. Un punto y coma provoca un error lógico y no sintáctico, por lo que el navegador no avisará.

Ejemplo:

```
for ( i = 0; i < 9; i++) {
   //instrucciones
}
```

Descripción: primero se escribe la palabra for, a continuación y entre paréntesis se escriben los tres campos: primero la acción inicial, en este caso es i = 0; después la condición para que se realicen las instrucciones, o sea, mientras i sea menor que 9 (i < 9) se efectuará; y por último el paso a siguiente vuelta (i++). Las instrucciones a repetir pueden incluir prácticamente lo que sea, puede ser funciones o lo que desees. Las partes del for van separadas por un punto y coma (**;**), pero las tres van entre paréntesis.
Las instrucciones a repetir van entre { }

Otro ejemplo:

```
for ( i=0; i<14; i++) {
alert ("El valor de i es igual a " + i )
}
```

En este ejemplo aparecerá, al iniciarse, un mensaje que indicará el valor actual de i, el mensaje se repetirá mientras el valor de i sea menor que 14 (i < 14)
o sea que el mensaje sera "El valor de i es igual a 0", "El valor de i es igual a 1", etc

While

La forma de escribir la instrucción while es:

```
while (condición){
// instrucciones
}
```

La condición va entre paréntesis y las instrucciones a repetir van entre { } (puede que no se realicen ninguna vez)
Ejemplo:

```
x = 0
while(x < 3) {
  x ++
}
```

En este ejemplo mientras (while) x sea menor que 3 (x<3) x aumentará en uno (x++), cuando x sea igual a 3 x dejará de aumentar.
Así seguirá ejecutándose hasta que la condición salga falsa. Es mas o menos como `for` pero en `while` sólo estableces la condición.

Funcionamiento del while: En primer lugar se comprueba la condición, si resulta verdadera se ejecutan las operaciones que indicamos entre las llaves que siguen al while. En caso de que la condición sea falsa continua con la instrucción siguiente al bloque de llaves. El bloque se repite MIENTRAS la condición sea Verdadera.
Importante: Si la condición siempre es verdadera estamos en presencia de un ciclo repetitivo infinito.

Ejemplo: Realizar un programa que escriba en la página los números del 1 al 100.

```
<html>
<head>
</head>
<body>
<script language="javascript">
var x;
x=1;
while (x<=100)
{
  document.write(x);
  document.write('<br>');
  x++;
}
</script>
</body>
</html>
```

Para que se escriban los números, uno en cada línea, agregamos la marca HTML de
.

La primera operación inicializa la variable x a 1, después comienza la estructura repetitiva while y tenemos la condición (x <= 100): se lee MIENTRAS la variable x sea menor o igual a 100.

Al comprobar la condición la primera vez sale VERDADERO, porque el contenido de x (1) es menor o igual a 100. Al ser la condición verdadera, se ejecuta el bloque de instrucciones que van entre llaves.

El bloque de instrucciones contiene dos salidas al documento y una operación. Se imprime el contenido de x y seguidamente se incrementa la variable x en uno. La operación x++ se lee como "en la variable x se guarda el contenido de x más 1". Es decir, si x contiene 1 después de ejecutarse esta operación se almacenará en x un 2.

Al finalizar el bloque de instrucciones, se comprueba nuevamente la condición y se repite el proceso explicado anteriormente.

Mientras la condición sea verdadera, se ejecuta el bloque de instrucciones; al ser falsa la verificación de la condición, se sale de la estructura repetitiva y continúa después; en este caso, finaliza el programa.

Lo más difícil puede ser la definición de la condición de la estructura while y qué bloque de instrucciones se va a repetir. Observa que si, por ejemplo, disponemos la condición x >=100 (si x es mayor o igual a 100) no provoca ningún error sintáctico (eso significa algo) pero estamos en presencia de un error lógico (no significa lo que nosotros queríamos) porque al comprobar por primera vez la condición sale falsa y no se ejecuta el bloque de instrucciones que queríamos repetir 100 veces.

No existe una RECETA para definir una condición de una estructura repetitiva, sino que se logra con una práctica continua, resolviendo problemas.

Si hacemos un seguimiento de los valores que toma x a lo largo de la ejecución tenemos:

```
x
1
2
3
4
.
.
100
101     Cuando x vale 101 la condición de la estructura
        repetitiva sale falsa, en este caso finaliza.
```

En este tipo de utilización, la variable x recibe el nombre de CONTADOR. Un contador es una variable que se incrementa o decrementa con valores constantes durante la ejecución del programa.
El contador x nos indica en cada momento la cantidad de valores escritos en la página.
La variable x debe estar inicializada con algún valor.
Se sugiere probar a modificar este programa y ver qué cambios se deberían hacer para:

1. Escribir los números del 1 al 500.
2. Escribir los números del 50 al 100.
3. Escribir los números del -50 al 0.
4. Escribir los números del 2 al 100 pero de 2 en 2 (2,4,6,8 ....100).

Do/while

La sentencia do/while es otra estructura repetitiva, la cual ejecuta siempre al menos una vez su bloque repetitivo, a diferencia del while que puede no ejecutar el bloque.

```
do(condición) {
acciones
} while(condición)
```

Esta estructura repetitiva se utiliza cuando conocemos de antemano que por lo menos una vez habrá que ejecutar el bloque repetitivo.

La condición de la estructura está al final del bloque a repetir, a diferencia del while que la lleva en la parte superior.

Finaliza la ejecución del bloque repetitivo cuando la condición resulta falsa, es decir igual que el while.

Ejemplo: Escribir un programa que solicite la carga de un número entre 0 y 999, y nos muestre un mensaje de cuántos dígitos tiene el mismo. Finalizar el programa cuando se cargue el valor 0.

```
<html>
<head>
</head>
<body>
```

```
<script language="javascript">
var valor;
do {
  valor=prompt('Valor entre 0 y 999:','');
  valor=parseInt(valor);
  document.write('El valor '+valor+' tiene ');
  if (valor<10)
  {
    document.write('1 dígito');
  }
  else
  {
    if (valor<100)
    {
      document.write('2 dígitos');
    }
    else
    {
      document.write('3 dígitos');
    }
  }
  document.write('<br>');
} while(valor!=0);
</script>
</body>
</html>
```

En este problema por lo menos se carga un valor. Si se carga un valor menor que 10 se trata de un número de una cifra, si es mayor que 10 pero menor que 100 se trata de un valor de dos dígitos, en caso contrario se trata de un valor de tres dígitos. Este bloque se repite mientras se meta en la variable 'valor' un número distinto de 0.

Otras instrucciones relacionadas con ciclos

Con la sentencia "break", se sale de un bloque de bucle o repetición.

Con la sentencia "continue", se termina la repetición actual y se comienza con la siguiente.

Ejemplo

Problema: Desarrollar un programa que permita la carga de 5 valores por teclado y nos muestre posteriormente la suma.

```
<html>
<head>
</head>
<body>
<script language="javascript">
var x=1;
var suma=0;
var valor;
while (x<=5)
{
  valor=prompt('Valor:','');
  valor=parseInt(valor);
  suma=suma+valor;
  x=x+1;
}
document.write("La suma de los valores es "+suma+"<br>");
</script>
</body>
```

```
</html>
```

En este problema, a semejanza de los anteriores, llevamos un CONTADOR llamado x que nos sirve para contar las vueltas que debe repetir el while.
También aparece un ACUMULADOR (un acumulador es una variable que usamos para incrementarla o decrementarla con valores variables durante la ejecución del programa).
Hemos dado el nombre de `suma` a nuestro acumulador. Cada ciclo que se repita la estructura repetitiva, la variable `suma` se incrementa con el contenido ingresado en la variable `valor`.

En un ejemplo podrían salir los siguietnes valores de las variables:

```
valor           suma            x
0               0               1(Antes de entrar a la estructura repetitiva).
5               5               2
16              21              3
7               28              4
10              38              5
2               40              6
```

Este es un seguimiento del programa planteado. Los números que toma la variable valor dependerá de qué cifras cargue el usuario durante la ejecución del programa.
Hay que tener en cuenta que cuando en la variable valor se carga el primer valor (en este ejemplo es 5), al cargarse el segundo valor (16), el valor anterior 5 se pierde, por ello es necesario ir almacenando en la variable `suma` el total de los valores cargados.

Funciones

Una función es una parte del programa creada con la finalidad de realizar una determinada acción. Una función puede ser llamada desde otra.

La forma de escribirlo es:

```
function nombreFunción(parámetros) {
acciones
}
```

Explicaremos con un ejemplo, una función que tiene datos de entrada.
Ejemplo: Confeccionar una función que reciba dos números y muestre en la página los valores comprendidos entre ellos de uno en uno. Cargar por teclado esos dos valores.

```
<html>
<head>
</head>
<body>
<script language="javascript">
function mostrarComprendidos(x1,x2)
{
  var inicio;
  for(inicio=x1;inicio<=x2;inicio++)
  {
    document.write(inicio+' ');
  }
}

var valor1,valor2;
valor1=prompt('Valor inferior:','');
valor1=parseInt(valor1);
valor2=prompt('Valor superior:','');
valor2=parseInt(valor2);
mostrarComprendidos(valor1,valor2);
</script>
```

```
</body>
</html>
```

El programa de JavaScript empieza a ejecutarse donde definimos las variables valor1 y valor2 y no donde se define la función, es decir, empieza fuera de cualquier función. Luego de cargar los dos valores por teclado se llama a la función `mostrarComprendidos` y le enviamos las variables valor1 y valor2. Los parámetros x1 y x2 reciben los contenidos de las variables valor1 y valor 2.

Si una función da un resultado con `return`, la función debe ser asignada a una variable, o formar parte de una respuesta. En caso contrario, como en el ejemplo, si una función no devuelve un valor con "return", puede ser llamada sin ser asignada.

Los argumentos de las funciones también se pueden manejar con un array propio de cada una de ellas. Al array se accede con "nombreFunción.arguments[i]", donde "i" es un índice que comienza por 0.

Para conocer el número de parámetros, podemos utilizar: "arguments.length".

A la función la podemos llamar la cantidad de veces que la necesitemos.

Otro ejemplo: si queremos que se muestre un mensaje de alerta cada vez que el usuario da click en un hipervínculo, ponemos el mensaje de alerta dentro de una función y luego en el hipervínculo escribimos el nombre de la función para que se active cada vez que demos click.

```
<script language="Javascript">
<!--
function mensaje( ){
alert("Mensaje de alerta.....")
}
// -->
</script>

<a href="javascript: mensaje( )">Da
click para un mensaje de alerta</a>
```

Para llamar a la función en este caso hemos creado un hipervínculo, primero con la palabra javascript, luego dos puntos : y por último el nombre de la función junto con los paréntesis ( ), es decir, javascript : mensaje ( )

Eventos

Evento se considera cualquier acción que el usuario realiza con el sistema: hacer click, posicionarse con el ratón en un lugar determinado, enviar un formulario, posicionarse en un cuadro para texto,....

Usaremos los eventos para activar nuestras funciones JavaScript. Para referirnos a un evento en HTML, el nombre del evento irá precedido por "on". Por ejemplo, el gestor de eventos de "Click" será "onClick".

Los eventos más habituales son los siguientes.

Nombre	Manejador	Descripción
Change	onChange	Un elemento de un formulario cambia de contenido
Click	onClick	El usuario hace click con el mouse sobre un formulario, una imagen, un link o cualquier elemento
keypress	onKeypress	El usuario presiona una tecla
load	onLoad	La página termina de cargarse completamente
mousemove	onMouseMove	El usuario mueve el cursor

mouseover	onmouseOver	El cursor está sobre un elemento
Select	onSelect	Se selecciona una de las opciones de un cuadro combo del formulario.
unload	onUnload	El usuario sale del documento

Para establecer la acción se pone el nombre del manejador, un signo de igual (=) y después el nombre de la función entre comillas (" "), ejemplo:

```
<script language="javascript"><!--
function aviso(){
alert ("El puntero esta sobre el link")
}
--></script>

<a href="#" onmouseover="aviso()">Link de prueba </a>
```

El evento de pasar el ratón sobre el botón es llamado con el manejador onmouseover

Puedes poner todos los manejadores que desees. Si quieres que se activen dos funciones separa el nombre de ellas con una coma y pon las dos entre comillas.
Otro ejemplo: Imaginemos que tenemos un botón en un formulario, y queremos que al pulsarlo realice una acción determinada:

```
<form ....>
<input type="button" onClick="función(parámetros)">
</form ....>
```

Con esta acción, asociamos al evento click sobre el botón, las acciones que realice la función.

Recuerda que para llamar a los eventos, se debe anteponer "on" al nombre del evento.

Sin embargo en jQuery, se asignan funciones a eventos sin necesidad del on. Por ejemplo, para que todos los párrafos al recibir una pulsación pasen a la función pulsa:

```
    $("p").click(pulsa);
```

En vez de click podría ser change, keypress, mouseover, select, ...