# Variables

Una variable es un depósito donde hay un valor. Consta de un nombre y pertenece a un tipo (numérico, cadena de caracteres, etc.)

Tipos de variable:

Una variable puede almacenar:

Valores Enteros (100, 260, etc.)
Valores Reales (1.24, 2.90, 5.01, etc.)
Cadenas de caracteres ('Juan', 'Compras', 'Listado', etc.)
Valores lógicos (true, false)

Existen otros tipos de variables que veremos más adelante.

Las variables son nombres que ponemos a los lugares donde almacenamos la información. En JavaScript, deben comenzar por una letra o un subrayado (_), pudiendo haber además dígitos entre los demás caracteres. Una variable no puede tener el mismo nombre de una palabra clave del lenguaje.

Una variable se define anteponiéndole la palabra clave var:
var dia;

se pueden declarar varias variables en una misma línea:
var dia, mes, anio;

a una variable se la puede definir e inmediatamente inicializarla con un valor:
var edad=20;

o en su defecto en dos pasos:
var edad;
edad=20;

Elección del nombre de una variable:

Debemos elegir nombres de variables representativos. En el ejemplo anterior los nombres dia, mes, anio son lo suficientemente claros para darnos una idea acabada sobre su contenido, una mala elección de nombres hubiera sido llamarlas a,b y c. Podemos darle otros buenos nombres. Otros no son tan representativos, por ejemplo d, m, a. Posiblemente cuando estemos resolviendo un problema dicho nombre nos recuerde que almacenamos el dia, pero pasado un tiempo lo olvidaríamos.

Impresión de variables en una página HTML.

Para mostrar el contenido de una variable en una página utilizamos el objeto document y llamamos a la función write.

En el siguiente ejemplo definimos una serie de variables y las mostramos en la página:

```
<html>
   <head>
    </head>
         <body>
             <script type="text/javascript">
            var nombre='Juan';
            var edad=10;
            var altura=1.92;
            var casado=false;
            document.write(nombre);
            document.write('<br>');
            document.write(edad);
            document.write('<br>');
            document.write(altura);
            document.write('<br>');
            document.write(casado);
             </script>
          </body>
</html>
```

Cuando imprimimos una variable, no la debemos disponer entre simples comillas (en caso de hacer esto, aparecerá el nombre de la variable y no su contenido)

Los valores de las variables que almacenan nombres (es decir, son cadenas de caracteres) deben ir encerradas entre comillas simples o dobles. Los valores de las variables enteras (en este ejemplo la variable edad) y reales no deben ir encerradas entre comillas. Cada instrucción finaliza con un punto y coma.

Las variables de tipo boolean pueden almacenar solo dos valores: true o false.
El resultado al visualizar la página debe ser 4 líneas similares a éstas:

```
Juan
10
1.92
false
```

Es decir que se muestran los contenidos de las 4 variables. Una variable es de un tipo determinado cuando le asignamos un valor:

```
var edad=10;
```

Es de tipo entera ya que le asignamos un valor entero.

```
var nombre='juan';
```

Es de tipo cadena.

Para mostrar el contenido de una variable en una página debemos utilizar la función 'write' que pertenece al objeto document. Recordemos que el lenguaje JavaScript es sensible a mayúsculas y minúsculas y no será lo mismo si escribimos:

```
Document.Write(nombre);
```

Esto porque no existe el objeto 'Document' sino el objeto 'document' (con d minúscula), lo mismo no existe la función 'Write' sino 'write', este es un error muy común cuando comenzamos a programar en JavaScript

## Entrada de datos por teclado.

Para la entrada de datos por teclado tenemos la función prompt. Cada vez que necesitamos ingresar un dato con esta función, aparece una ventana donde cargamos el valor. Hay otras formas más sofisticadas para la entrada de datos en una página HTML, pero para el aprendizaje de los conceptos básicos de JavaScript nos resultará más práctica esta función.

Para ver su funcionamiento analicemos este ejemplo:

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
  var nombre;
  var edad;
  nombre=prompt('Ingrese su nombre:','');
  edad=prompt('Ingrese su edad:','');
  document.write('Hola ');
  document.write(nombre);
  document.write(' asi que tienes ');
  document.write(edad);
  document.write(' años');
</script>
</body>
</html>
```

La sintaxis de la función prompt es:

```
<variable que recibe el dato>=prompt(<mensaje a mostrar en la
ventana>,<valor
  inicial a mostrar en la ventana>);
```

La función prompt tiene dos parámetros: uno es el mensaje y el otro el valor inicial a mostrar.

## Estructuras secuenciales de programación.

Cuando en un problema sólo participan operaciones, entradas y salidas se la denomina estructura secuencial.

El problema anterior, donde se ingresa el nombre de una persona y su edad se trata de una estructura secuencial.

Ejemplo de otro algoritmo con estructura secuencial: Realizar la carga de dos números por teclado e imprimir su suma y su producto:

```
<html>
<head>
<script type="text/javascript">
  var valor1;
  var valor2;
  valor1=prompt('Ingrese primer número:','');
  valor2=prompt('Ingrese segundo número','');
  var suma=parseInt(valor1)+parseInt(valor2);
  var producto=parseInt(valor1)*parseInt(valor2);
  document.write('La suma es ');
  document.write(suma);
  document.write('<br>');
  document.write('El producto es ');
  document.write(producto);
</script>
</head>
<body>
</body>
</html>
```

Lo primero que debemos tener en cuenta es que si queremos que el operador + sume los contenidos de los valores numéricos ingresados por teclado, debemos llamar a la función parseInt y pasar como parámetro las variables valor1 y valor2 sucesivamente. Con esto logramos que el operador más, sume las variables como enteros y no como cadenas de caracteres. Si por ejemplo sumamos 1 + 1 sin utilizar la función parseInt el resultado será 11 en lugar de 2, ya que el operador + concatena las dos cadenas.

En JavaScript, como no podemos indicarle de qué tipo es la variable, requiere mucho más cuidado cuando operamos con sus contenidos.

Este problema es secuencial ya que ingresamos dos valores por teclado, luego hacemos dos operaciones y por último mostramos los resultados.

## Estructuras condicionales simples.

No todos los problemas pueden resolverse empleando estructuras secuenciales. Cuando hay que tomar una decisión aparecen las estructuras condicionales.

En nuestra vida diaria se nos presentan situaciones donde debemos decidir.

¿Elijo la carrera A o la carrera B ?
¿Me pongo este pantalón ?
¿Entro al sitio A o al sitio B ?
Para ir al trabajo, ¿elijo el camino A o el camino B ?
Al cursar una carrera, ¿elijo el turno mañana, tarde o noche ?

Por supuesto que en un problema se combinan estructuras secuenciales y condicionales.

Cuando se presenta la elección tenemos la opción de realizar una actividad o no realizarla.

En una estructura CONDICIONAL SIMPLE por el camino del verdadero hay actividades y por el camino del falso no hay actividades. Por el camino del verdadero pueden existir varias operaciones, entradas y salidas, inclusive ya veremos que puede haber 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 4:

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
  var nombre;
  var nota;
  nombre=prompt('Ingrese nombre:','');
  nota=prompt('Ingrese su nota:','');
  if (nota>=4)
  {
    document.write(nombre+' esta aprobado con un '+nota);
  }
</script>
```

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

Aparece la instrucción if en el lenguaje JavaScript. La condición debe ir entre paréntesis. Si la condición se verifica verdadera se ejecuta todas las instrucciones que se encuentran encerradas entre las llaves de apertura y cerrado seguidas al if.
Para disponer condiciones en un if podemos utilizar alguno de los siguientes operadores relacionales:

> mayor
>= mayor o igual
< menor
<= menor o igual
!= distinto
== igual

Siempre debemos tener en cuenta que en la condición del if deben intervenir una variable un operador relacional y otra variable o valor fijo.

Otra cosa que hemos incorporado es el operador + para cadenas de caracteres:
document.write(nombre+' esta aprobado con un '+nota);

Con esto hacemos más corto la cantidad de líneas de nuestro programa, recordemos que veníamos haciéndolo de la siguiente forma:

```
document.write(nombre);
document.write(' esta aprobado con un ');
document.write(nota);
```

## Estructuras condicionales compuestas.

Cuando se presenta la elección tenemos la opción de realizar una actividad u otra. Es decir tenemos actividades por el verdadero y por el falso de la condición. Lo más importante que hay que tener en cuenta es que se realizan las actividades de la rama del verdadero o las del falso, NUNCA se realizan las actividades de las dos ramas.
En una estructura condicional compuesta tenemos entradas, salidas, operaciones, tanto por la rama del verdadero como por la rama del falso.

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

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

```
<script type="text/javascript">
  var num1,num2;
  num1=prompt('Ingrese el primer número:','');
  num2=prompt('Ingrese el 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 string por lo que debemos convertirlo a entero cuando queremos saber cual de los dos valores es mayor numéricamente. En el lenguaje JavaScript una variable puede ir cambiando el tipo de dato que almacena a lo largo de la ejecución del programa.

Más adelante veremos qué sucede cuando preguntamos cuál de dos string es mayor.

Estamos en presencia de una ESTRUCTURA CONDICIONAL COMPUESTA ya que tenemos actividades por la rama del verdadero y del falso.

La estructura condicional compuesta tiene la siguiente codificación:

```
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 que la condición resulte falsa se ejecuta la instrucción o bloque de instrucciones que indicamos después del else.

# Estructuras condicionales anidadas.

Decimos que una estructura condicional es anidada cuando por la rama del verdadero o el falso de una estructura condicional hay otra estructura condicional.

Ejemplo: Confeccionar un programa que pida por teclado tres notas de un alumno, calcule el promedio e imprima alguno de estos mensajes:

Si el promedio es >=7 mostrar "Promocionado".
Si el promedio es >=4 y <7 mostrar "Regular".
Si el promedio es <4 mostrar "Reprobado".

Solución:

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
  var nota1,nota2,nota3;
  nota1=prompt('Ingrese 1ra. nota:','');
  nota2=prompt('Ingrese 2da. nota:','');
  nota3=prompt('Ingrese 3ra. nota:','');
  //Convertimos los 3 string en enteros
  nota1=parseInt(nota1);
  nota2=parseInt(nota2);
  nota3=parseInt(nota3);
  var pro;
  pro=(nota1+nota2+nota3)/3;
  if (pro>=7)
  {
    document.write('promocionado');
  }
  else
  {
    if (pro>=4)
    {
      document.write('regular');
    }
    else
    {
      document.write('reprobado');
    }
  }
</script>
</body>
</html>
```

Analicemos el siguiente programa. Se ingresan tres string por teclado que representan las notas de un alumno, se transformas a variables enteras y se obtiene el promedio sumando los tres valores y dividiendo por 3 dicho resultado.

Primeramente preguntamos si el promedio es superior o igual a 7, en caso afirmativo por la rama del verdadero de la estructura condicional mostramos un mensaje que indique 'Promocionado' (con comillas indicamos un texto que debe imprimirse en pantalla).

En caso que la condición nos de falso, por la rama del falso aparece otra estructura condicional, porque todavía debemos averiguar si el promedio del alumno es superior o igual a cuatro o inferior a cuatro.

Los comentarios en JavaScript los hacemos disponiendo dos barras previas al comentario (los comentario en tiempo de ejecución no son tenidos en cuenta y tienen por objetivos de documentar el programa para futuras modificaciones):

//Convertimos los 3 string en enteros

Si queremos disponer varias líneas de comentarios tenemos como alternativa:

```
/*
linea de comentario 1.
linea de comentario 2.
etc.
*/
```

Es decir encerramos el bloque con los caracteres `/* */`

## Operadores lógicos && (y) en las estructuras condicionales.

El operador &&, traducido se lo lee como "Y". Se emplea cuando en una estructura condicional se disponen dos condiciones.

Cuando vinculamos dos o más condiciones con el operador "&&" las dos condiciones deben ser verdaderas para que el resultado de la condición compuesta de Verdadero y continúe por la rama del verdadero de la estructura condicional.

Recordemos que la condición debe ir entre paréntesis en forma obligatoria.

La utilización de operadores lógicos permiten en muchos casos, plantear algoritmos más cortos y comprensibles.

Veamos un ejemplo: Confeccionar un programa que lea por teclado tres números distintos y nos muestre el mayor de ellos.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
  var num1,num2,num3;
  num1=prompt('Ingrese primer número:','');
  num2=prompt('Ingrese segundo número:','');
  num3=prompt('Ingrese tercer número:','');
  num1=parseInt(num1);
  num2=parseInt(num2);
  num3=parseInt(num3);
  if (num1>num2 && num1>num3)
  {
    document.write('el mayor es el '+num1);
  }
  else
  {
    if (num2>num3)
    {
      document.write('el mayor es el '+num2);
    }
    else
    {
      document.write('el mayor es el '+num3);
    }
  }
</script>
</body>
</html>
```

Podemos leerla de la siguiente forma:

Si el contenido de la variable num1 es mayor al contenido de la variable num2 Y si el contenido de la variable num1 es mayor al contenido de la variable num3 entonces la CONDICION COMPUESTA resulta Verdadera.

Si una de las condiciones simples da falso, la CONDICION COMPUESTA da Falso y continúa por la rama del falso.

Es decir que se mostrará el contenido de num1 si y sólo si num1>num2 y num1>num3.

En caso de ser Falsa la condición de la rama del falso, analizamos el contenido de num2 y num3 para ver cual tiene un valor mayor.

En esta segunda estructura condicional, al haber una condición simple, no se requieren operadores lógicos.

## Operadores lógicos || (o) en las estructuras condicionales.

Traducido se lo lee como "O". Si la condición 1 es Verdadera o la condición 2 es Verdadera, luego ejecutar la rama del Verdadero.

Cuando vinculamos dos o más condiciones con el operador "O", con que una de las dos condiciones sea Verdadera alcanza para que el resultado de la condición compuesta sea Verdadero.

Ejemplo: Se carga una fecha (día, mes y año) por teclado. Mostrar un mensaje si corresponde al primer trimestre del año (enero, febrero o marzo).

Cargar por teclado el valor numérico del día, mes y año por separado.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
  var dia,mes,año;
  dia=prompt('Ingrese día:','');
  mes=prompt('Ingrese mes:','');
  año=prompt('Ingrese año:','');
  dia=parseInt(dia);
  mes=parseInt(mes);
  año=parseInt(año);
  if (mes==1 || mes==2 || mes==3)
  {
    document.write('corresponde al primer trimestre del
año.');
  }
</script>
</body>
</html>
```

La carga de una fecha se hace por partes, ingresamos las variables dia, mes y año.

# Estructuras switch.

La instrucción switch es una alternativa para remplazar los if/else if.

De todos modos se puede aplicar en ciertas situaciones donde la condición se verifica si es igual a cierto valor. No podemos preguntar por mayor o menor.

Con un ejemplo sencillo veremos cual es su sintaxis. Confeccionar un programa que solicite que ingrese un valor entre 1 y 5. Luego mostrar en castellano el valor ingresado. Mostrar un mensaje de error en caso de haber ingresado un valor que no se encuentre en dicho rango.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
  var valor;
  valor=prompt('Ingrese un valor comprendido entre 1 y 5:','');
  //Convertimos a entero
  valor=parseInt(valor);
  switch (valor) {
    case 1: document.write('uno');
            break;
    case 2: document.write('dos');
            break;
    case 3: document.write('tres');
            break;
    case 4: document.write('cuatro');
            break;
    case 5: document.write('cinco');
            break;
    default:document.write('debe ingresar un valor comprendido
entre 1 y 5.');
  }
</script>
</body>
</html>
```

Debemos tener en cuenta que la variable que analizamos debe ir después de la instrucción switch entre paréntesis. Cada valor que se analiza debe ir luego de la palabra clave 'case' y seguido a los dos puntos, las instrucciones a ejecutar, en caso de verificar dicho valor la variable que analiza el switch.

Es importante disponer la palabra clave 'break' al finalizar cada caso. La instrucciones que hay después de la palabra clave 'default' se ejecutan en caso que

la variable no se verifique en algún case. De todos modos el default es opcional en esta instrucción.

Plantearemos un segundo problema para ver que podemos utilizar variables de tipo cadena con la instrucción switch.

Ingresar por teclado el nombre de un color (rojo, verde o azul), luego pintar el fondo de la ventana con dicho color:

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
  var col;
  col=prompt('Ingrese el color con que se quiere pintar el
fondo de la ventana (rojo, verde, azul)' ,'');
  switch (col) {
    case 'rojo': document.bgColor='#ff0000';
                 break;
    case 'verde': document.bgColor='#00ff00';
                 break;
    case 'azul': document.bgColor='#0000ff';
                 break;
  }
</script>
</body>
</html>
```

Cuando verificamos cadenas debemos encerrarlas entre comillas el valor a analizar:

```
  case 'rojo': document.bgColor='#ff0000';
               break;
```

Para cambiar el color de fondo de la ventana debemos asignarle a la propiedad bgColor del objeto document el color a asignar (el color está formado por tres valores hexadecimales que representan la cantidad de rojo, verde y azul), en este caso al valor de rojo le asignamos ff (255 en decimal) es decir el valor máximo posible, luego 00 para verde y azul (podemos utilizar algún software de graficación para que nos genere los tres valores).

## Estructura repetitiva (while)

Hasta ahora hemos empleado estructuras SECUENCIALES y CONDICIONALES. Existe otro tipo de estructuras tan importantes como las anteriores que son las estructuras REPETITIVAS.

Una estructura repetitiva permite ejecutar una instrucción o un conjunto de instrucciones varias veces.

Una ejecución repetitiva de sentencias se caracteriza por:

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

Funcionamiento del while: En primer lugar se verifica la condición, si la misma resulta verdadera se ejecutan las operaciones que indicamos entre las llaves que le siguen al while.

En caso que la condición sea Falsa continúa con la instrucción siguiente al bloque de llaves.

El bloque se repite MIENTRAS la condición sea Verdadera.

Importante: Si la condición siempre retorna verdadero estamos en presencia de un ciclo repetitivo infinito. Dicha situación es un error de programación, nunca finalizará el programa.

Ejemplo: Realizar un programa que imprima en pantalla los números del 1 al 100.

Sin conocer las estructuras repetitivas podemos resolver el problema empleando una estructura secuencial. Inicializamos una variable con el valor 1, luego imprimimos la variable, incrementamos nuevamente la variable y así sucesivamente. Pero esta solución es muy larga.

La mejor forma de resolver este problema es emplear una estructura repetitiva:

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

Para que se impriman los números, uno en cada línea, agregamos la marca HTML de <br>.

Es muy importante analizar este programa:

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

Al ejecutarse la condición, retorna 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 contiene la estructura while. 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 = x + 1 se lee como "en la variable x se guarda el contenido de x más 1". Es decir, si x contiene 1 luego de ejecutarse esta operación se almacenará en x un 2.

Al finalizar el bloque de instrucciones que contiene la estructura repetitiva, se verifica nuevamente la condición de la estructura repetitiva y se repite el proceso explicado anteriormente.

Mientras la condición retorne verdadero, se ejecuta el bloque de instrucciones; al retornar falso la verificación de la condición, se sale de la estructura repetitiva y continúa el algoritmo, en este caso, finaliza el programa.

Lo más difícil es la definición de la condición de la estructura while y qué bloque de instrucciones se va a repetir. Observar que si, por ejemplo, disponemos la condición x >=100 ( si x es mayor o igual a 100) no provoca ningún error sintáctico pero estamos en presencia de un error lógico porque al evaluarse por primera vez la condición retorna falso 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, solucionando problemas.

Una vez planteado el programa debemos verificar si el mismo es una solución válida al problema (en este caso se deben imprimir los números del 1 al 100 en la página), para ello podemos hacer un seguimiento del flujo del diagrama y los valores que toman las variables a lo largo de la ejecución:

```
     x
     1
     2
     3
     4
     .
     .
     100
        101    Cuando x vale 101 la condición de la estructura
                repetitiva retorna falso, en este caso
finaliza el diagrama.
```

La variable x recibe el nombre de CONTADOR. Un contador es un tipo especial de 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 impresos en la página.

Importante: Podemos observar que el bloque repetitivo puede no ejecutarse si la condición retorna falso la primera vez.

La variable x debe estar inicializada con algún valor antes que se ejecute la operación x = x + 1.

Probemos algunas modificaciones de este programa y veamos qué cambios se deberían hacer para:

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

## Concepto de acumulador.

Explicaremos el concepto de un acumulador con un 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  type="text/javascript">
  var x=1;
  var suma=0;
  var valor;
```

```
  while (x<=5)
  {
    valor=prompt('Ingrese 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 el concepto de ACUMULADOR (un acumulador es un tipo especial de variable que se incrementa o decrementa 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.

La prueba del diagrama se realiza dándole valores a las variables:

```
valor          suma      x
0         0
```

(Antes de entrar a la estructura repetitiva estos son los valores).

```
5         5         1
16        21        2
7         28        3
10        38        4
2         40        5
```

Este es un seguimiento del programa planteado. Los números que toma la variable valor dependerá de qué cifras cargue el operador 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 el valor 5), al cargarse el segundo valor (16), el valor anterior 5 se pierde, por ello la necesidad de ir almacenando en la variable suma el valor acumulado de los valores ingresados.

## Estructura repetitiva (do/while)

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

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

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

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

Problema: 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  type="text/javascript">
  var valor;
  do {
    valor=prompt('Ingrese un valor entre 0 y 999:','');
    valor=parseInt(valor);
    document.write('El valor '+valor+' tiene ');
    if (valor<10)
    {
      document.write('Tiene 1 digitos');
    }
    else
    {
      if (valor<100)
      {
        document.write('Tiene 2 digitos');
      }
      else
      {
        document.write('Tiene 3 digitos');
      }
    }
   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 a 10 se trata de un número de una cifra, si es mayor a 10 pero menor a 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 ingresa en la variable 'valor' un número distinto a 0.

## Estructura repetitiva (for)

Cualquier problema que requiera una estructura repetitiva se puede resolver empleando la estructura while. Pero hay otra estructura repetitiva cuyo planteo es más sencillo en ciertas situaciones.

Esta estructura se emplea en aquellas situaciones en las cuales CONOCEMOS la cantidad de veces que queremos que se ejecute el bloque de instrucciones. Ejemplo: cargar 10 números, ingresar 5 notas de alumnos, etc. Conocemos de antemano la cantidad de veces que queremos que el bloque se repita.

Por último, hay que decir que la ejecución de la sentencia break dentro de cualquier parte del bucle provoca la salida inmediata del mismo.

Sintaxis:

```
for (<Inicialización> ; <Condición> ; <Incremento o Decremento>)
{
  <Instrucciones>
}
```

Esta estructura repetitiva tiene tres argumentos: variable de inicialización, condición y variable de incremento o decremento.

Funcionamiento:

- Primero se ejecuta por única vez el primer argumento .
  Por lo general se inicializa una variable.
- El segundo paso es evaluar la (Condición), en caso de ser verdadera se ejecuta el bloque, en caso contrario continúa el programa.
- El tercer paso es la ejecución de las instrucciones.
- El cuarto paso es ejecutar el tercer argumento (Incremento o Decremento).
- Luego se repiten sucesivamente del Segundo al Cuarto Paso.

Este tipo de estructura repetitiva se utiliza generalmente cuando sabemos la cantidad de veces que deseamos que se repita el bloque.

Ejemplo: Mostrar por pantalla los números del 1 al 10.

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

<script type="text/javascript">
  var f;
  for(f=1;f<=10;f++)
  {
    document.write(f+" ");
  }
</script>

</body>
</html>
```

Inicialmente f se la inicializa con 1. Como la condición se verifica como verdadera se ejecuta el bloque del for (en este caso mostramos el contenido de la variable f y un espacio en blanco). Luego de ejecutar el bloque pasa al tercer argumento del for (en este caso con el operador ++ se incrementa en uno el contenido de la variable f, existe otro operador -- que decrementa en uno una variable), hubiera sido lo mismo poner f=f+1 pero este otro operador matemático nos simplifica las cosas.

Importante: Tener en cuenta que no lleva punto y coma al final de los tres argumentos del for. El disponer un punto y coma provoca un error lógico y no sintáctico, por lo que el navegador no avisará.

# Funciones

En programación es muy frecuente que un determinado procedimiento de cálculo definido por un grupo de sentencias tenga que repetirse varias veces, ya sea en un mismo programa o en otros programas, lo cual implica que se tenga que escribir tantos grupos de aquellas sentencias como veces aparezca dicho proceso.

La herramienta más potente con que se cuenta para facilitar, reducir y dividir el trabajo en programación, es escribir aquellos grupos de sentencias una sola y única vez bajo la forma de una FUNCION.

Un programa es una cosa compleja de realizar y por lo tanto es importante que esté bien ESTRUCTURADO y también que sea inteligible para las personas. Si un grupo de sentencias realiza una tarea bien definida, entonces puede estar justificado el aislar estas sentencias formando una función, aunque resulte que sólo se le llame o use una vez.

Hasta ahora hemos visto como resolver un problema planteando un único algoritmo. Con funciones podemos segmentar un programa en varias partes.

Frente a un problema, planteamos un algoritmo, éste puede constar de pequeños algoritmos.

Una función es un conjunto de instrucciones que resuelven una parte del problema y que puede ser utilizado (llamado) desde diferentes partes de un programa.

Consta de un nombre y parámetros. Con el nombre llamamos a la función, es decir, hacemos referencia a la misma. Los parámetros son valores que se envían y son indispensables para la resolución del mismo. La función realizará alguna operación con los parámetros que le enviamos. Podemos cargar una variable, consultarla, modificarla, imprimirla, etc.

Incluso los programas más sencillos tienen la necesidad de fragmentarse. Las funciones son los únicos tipos de subprogramas que acepta JavaScript. Tienen la siguiente estructura:

```
function <nombre de función>(argumento1, argumento2, ..., argumento n)
{
  <código de la función>
}
```

Debemos buscar un nombre de función que nos indique cuál es su objetivo (Si la función recibe un string y lo centra, tal vez deberíamos llamarla centrarTitulo). Veremos que una función puede variar bastante en su estructura, puede tener o no parámetros, retornar un valor, etc.

Ejemplo: Mostrar un mensaje que se repita 3 veces en la página con el siguiente texto:

'Cuidado'
'Ingrese su documento correctamente'

'Cuidado'
'Ingrese su documento correctamente'

'Cuidado'
'Ingrese su documento correctamente'

La solución sin emplear funciones es:

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

<script type="text/javascript">
  document.write("Cuidado<br>");
  document.write("Ingrese su documento correctamente<br>");
  document.write("Cuidado<br>");
  document.write("Ingrese su documento correctamente<br>");
  document.write("Cuidado<br>");
  document.write("Ingrese su documento correctamente<br>");
</script>

</body>
</html>
Empleando una función:
<html>
<head>
</head>
<body>

<script type="text/javascript">
  function mostrarMensaje()
  {
    document.write("Cuidado<br>");
    document.write("Ingrese su documento correctamente<br>");
  }

  mostrarMensaje();
  mostrarMensaje();
  mostrarMensaje();
</script>

</body>
</html>
```

Recordemos que JavaScript es sensible a mayúsculas y minúsculas. Si fijamos como nombre a la función mostrarTitulo (es decir la segunda palabra con mayúscula) debemos respetar este nombre cuando la llamemos a dicha función.

Es importante notar que para que una función se ejecute debemos llamarla desde fuera por su nombre (en este ejemplo: mostrarMensaje()).

Cada vez que se llama una función se ejecutan todas las líneas contenidas en la misma.

Si no se llama a la función, las instrucciones de la misma nunca se ejecutarán.

A una función la podemos llamar tantas veces como necesitemos.

Las funciones nos ahorran escribir código que se repite con frecuencia y permite que nuestro programa sea más entendible.

## Funciones con parámetros.

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 type="text/javascript">

  function mostrarComprendidos(x1,x2)
  {
    var inicio;
    for(inicio=x1;inicio<=x2;inicio++)
    {
      document.write(inicio+' ');
    }
  }

  var valor1,valor2;
  valor1=prompt('Ingrese valor inferior:','');
  valor1=parseInt(valor1);
  valor2=prompt('Ingrese 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. 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 valor2.

Es importante notar que a la función la podemos llamar la cantidad de veces que la necesitemos.

Los nombres de los parámetros, en este caso se llaman x1 y x2, no necesariamente se deben llamar igual que las variables que le pasamos cuando la llamamos a la función, en este caso le pasamos los valores valor1 y valor2.

## Funciones que retornan un valor.

Son comunes los casos donde una función, luego de hacer un proceso, retorne un valor.

Ejemplo 1: Confeccionar una función que reciba un valor entero comprendido entre 1 y 5. Luego retornar en castellano el valor recibido.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">

  function convertirCastellano(x)
  {
    if (x==1)
      return "uno";
    else
      if (x==2)
        return "dos";
      else
        if (x==3)
          return "tres";
        else
          if (x==4)
            return "cuatro";
          else
            if (x==5)
              return "cinco";
            else
              return "valor incorrecto";
  }

  var valor;
  valor=prompt("Ingrese un valor entre 1 y 5","");
  valor=parseInt(valor);
```

```
  var r;
  r=convertirCastellano(valor);
  document.write(r);

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

Podemos ver que el valor retornado por una función lo indicamos por medio de la palabra clave return. Cuando se llama a la función, debemos asignar el nombre de la función a una variable, ya que la misma retorna un valor.

Una función puede tener varios parámetros, pero sólo puede retornar un único valor. La estructura condicional if de este ejemplo puede ser remplazada por la instrucción switch, la función queda codificada de la siguiente manera:

```
function convertirCastellano(x)
{
  switch (x)
  {
    case 1:return "uno";
    case 2:return "dos";
    case 3:return "tres";
    case 4:return "cuatro";
    case 5:return "cinco";
    default:return "valor incorrecto";
  }
}
```

Esta es una forma más elegante que una serie de if anidados. La instrucción switch analiza el contenido de la variable x con respecto al valor de cada caso. En la situación de ser igual, ejecuta el bloque seguido de los 2 puntos hasta que encuentra la instrucción return o break.

Ejemplo 2: Confeccionar una función que reciba una fecha con el formato de día, mes y año y retorne un string con un formato similar a: "Hoy es 10 de junio de 2013".

```
<html>
<head>
</head>
<body>
<script type="text/javascript">

  function formatearFecha(dia,mes,año)
  {
    var s='Hoy es '+dia+' de ';
    switch (mes) {
```

```
    case 1:s=s+'enero ';
           break;
    case 2:s=s+'febrero ';
           break;
    case 3:s=s+'marzo ';
           break;
    case 4:s=s+'abril ';
           break;
    case 5:s=s+'mayo ';
           break;
    case 6:s=s+'junio ';
           break;
    case 7:s=s+'julio ';
           break;
    case 8:s=s+'agosto ';
           break;
    case 9:s=s+'septiembre ';
           break;
    case 10:s=s+'octubre ';
           break;
    case 11:s=s+'noviembre ';
           break;
    case 12:s=s+'diciembre ';
           break;
    } //fin del switch
    s=s+'de '+año;
    return s;
  }

  document.write(formatearFecha(11,6,2013));

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

Analicemos un poco la función formatearFecha. Llegan tres parámetros con el día, mes y año. Definimos e inicializamos una variable con:

```
   var s='Hoy es '+dia+' de ';
```

Luego le concatenamos o sumamos el mes:

```
s=s+'enero ';
```

Esto, si el parámetro mes tiene un uno. Observemos como acumulamos lo que tiene 's' más el string 'enero '. En caso de hacer s='enero ' perderíamos el valor previo que tenía la variable s.

Por último concatenamos el año:

```
s=s+'de '+año;
```

Cuando se llama a la función directamente, al valor devuelto se lo enviamos a la función write del objeto document. Esto último lo podemos hacer en dos pasos:

```
  var fec= formatearFecha(11,6,2013);
  document.write(fec);
```

Guardamos en la variable 'fec' el string devuelto por la función.