INGENIERIA WEB II
DISEÑO DE APLICACIONES WEB
INTRODUCCIÓN A JAVASCRIPT - Teórico y Actividades
DOCENTE: ING. CARLOS SIMES

Introducción a Javascript

JavaScript, al igual que Flash(tecnología obsoleta), Visual Basic Script (tecnología obsoleta), Applet (tecnología obsoleta), es una de las múltiples maneras que han surgido para extender las capacidades del lenguaje HTML (lenguaje para el diseño de páginas de Internet). Al ser la más sencilla, es por el momento la más extendida.

JavaScript no es un lenguaje de programación propiamente dicho como C, C++ etc. con el que puedo generar aplicaciones autónomas.

Es un lenguaje script u orientado a documento, como pueden ser los lenguajes de macros que tienen muchos procesadores de texto y planillas de cálculo. Hasta hace poco no se podía desarrollar programas con JavaScript que se ejecutaran fuera de un Navegador, aunque en este momento comienza a expandirse a otras áreas como la programación en el servidor con Node.js

JavaScript es un lenguaje interpretado que se embebe en una página web HTML. Un lenguaje interpretado significa que a las instrucciones las analiza y procesa el navegador en el momento que deben ser ejecutadas.

Nuestro primer programa será el famoso "Hola Mundo", es decir un programa que muestre en el documento HTML el mensaje "Hola Mundo".

Ejemplo01.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de JavaScript</title>
  <meta charset="UTF-8">
</head>
<body>

<script>
```

```
  document.write('Hola Mundo');
</script>
</body>
</html>
```

En este ejemplo estamos usando un script embebido, es decir las instrucciones de javascript se encuentran en la misma página. Más adelante veremos como crear archivos externos para colocar los scripts.

Vemos aquí una primera instrucción de javascript **document.write**

Esta instrucción permite escribir algo en el body de la página web.
Para ver el resultado del ejemplo cree una página html con el código del ejemplo y ábrala en el navegador para mostrar el resultado.

El programa en JavaScript debe ir encerrado entre las marcar 'script':

```
<script>
</script>
```

En versiones anteriores a HTML5 el programa en JavaScript debe ir encerrado entre la marca script e inicializada la propiedad type con la cadena text/javascript:

```
<script type="text/javascript">
</script>
```

La información a imprimirse debe ir entre comillas y encerrada entre paréntesis. Todo lo que indicamos entre comillas aparecerá tal cual dentro de la página HTML.
Es decir, si pedimos al navegador que ejecute esta página mostrará el texto 'Hola Mundo'.
Cada vez que escribimos una instrucción finalizamos con el carácter punto y coma.

ES IMPORTANTISIMO TENER EN CUENTA QUE JavaScript es SENSIBLE A MAYUSCULAS Y MINUSCULAS. NO ES LO MISMO ESCRIBIR: document.write que DOCUMENT.WRITE (la primera forma es la correcta, la segunda forma provoca un error de sintaxis).

Nos acostumbraremos a prestar atención cada vez que escribamos en minúsculas o mayúsculas para no cometer errores sintácticos. Ya veremos que los nombres de funciones llevan letras en mayúsculas.

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;

Javascript es un lenguaje débilmente tipeado, lo que significa que no es necesario definir el tipo de variable cuando la declaramos. El interprete asigna el tipo de variable en función del valor que le asignamos, por ejemplo:
var edad;

edad=26
hace que la variable edad se interprete como variable de tipo numérica, en cambio si hacemos
var edad;
edad="carlos"
hace que la variable edad se interprete como variable de tipo string.
Esto nos dá la ventaja a la hora de definir variables ya que no es necesario especificar de entrada el tipo de datos.
Además el interprete hace casting automático siempre que sea posible sin necesidad de especificar dicho casteo (convierte los tipos de variables en forma automática, siempre que sea posible).
Esto simplifica la programación, pero se debe estar atentos pues no siempre los resultados pueden ser los esperados.

Elección del nombre de una variable:

Debemos elegir nombres de variables representativos. En el ejemplo anterior los nombres día, 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 día, 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:

Ejemplo02.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de JavaScript</title>
```

```
  <meta charset="UTF-8">
</head>
<body>

<script>
  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 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 tipeamos:

```
Document.Write(nombre);
```
se produce un error

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

Observemos también que document.write nos permite incluir etiquetas html, lo que podemos usar para dar formato a lo que queremos mostrar en la página.

Por ejemplo document.write('<h1> Hola a todos </h1>'); mostrará en la página Hola a todos pero con formato de encabezado h1.

Entrada de datos por teclado

Para la entrada de datos por teclado podemos usar la función **prompt**.
Cada vez que necesitamos ingresar un dato con ésta 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:

Ejemplo03.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de JavaScript</title>
  <meta charset="UTF-8">
</head>
<body>

<script>
  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:

var dato = prompt (mensaje, 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.

Actividad Nº1:

Realice una página html que permita el ingreso de dos números y que muestre el valor de la suma de ambos números.

Si tienes problemas para resolver esta actividad, sigue adelante con los próximos temas donde se encuentra la misma resuelta y explicada.

Estructuras secuenciales de programación

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

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

Ejemplo04.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de JavaScript</title>
  <meta charset="UTF-8">
</head>
<body>

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

</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 '+', 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, se 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

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:

Ejemplo05.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de JavaScript</title>
  <meta charset="UTF-8">
</head>
<body>

<script>
  var nombre;
  var nota;
  nombre=prompt('Ingrese nombre:','');
```

```
 nota=parseInt(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.**
La instrucción if admite al igual que en otros idiomas la cláusula else y puede ser anidada.
La estructura quedaría **if { .....; } else { .....; }**

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.

Como queremos que en la variable 'nota' se guarde como entero lo convertimos llamando a parseInt:

```
 nota=parseInt(prompt('Ingrese su nota:',''));
```

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);
```

Actividad Nº2:

Usando lo aprendido hasta aquí, cree una página html con un script que permita el ingreso del Apellido de un alumno, permita ingresar 3 notas. Con esa información, calcularemos el promedio del alumno e indicaremos si está aprobado (promedio > = 6) o rinde en coloquio (promedio >=4 && promedio <6) o rinde en marzo (promedio <4)

Operadores Lógicos

El lenguaje javascript admite los siguientes operadores lógicos que pueden usarse para combinar distintas expresiones a evaluar:

&& operador and: evalua dos expresiones que deben ser ambas verdaderas, por ejemplo:
if ((A<5) && (B< 6)) { .....} aquí ambas comparaciones deben ser verdaderas para que se ejecuten las instrucciones dentro del if

|| operador or: evalua dos expresiones donde al menos una debe cumplirse como válida, por ejemplo:

if ((A<5) **||** (B< 6)) { .....} aquí al menos una de las comparaciones debe ser verdadera para que se ejecuten las instrucciones dentro del if

Estructuras switch

La instrucción switch es una alternativa para remplazar en algunas situaciones 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 cuál es su sintaxis. Confeccionar un programa que solicite que ingrese un valor entre 1 y 5. Luego mostrar en texto el valor ingresado. Mostrar un mensaje de error en caso de haber ingresado un valor que no se encuentre en dicho rango.

Ejemplo06.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de JavaScript</title>
  <meta charset="UTF-8">
</head>
```

```
<body>

<script>
  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 mostraremos un mensaje indicando el color ingresado:

Ejemplo07.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de JavaScript</title>
  <meta charset="UTF-8">
</head>
<body>

<script>
  var color;
  color=prompt('Ingrese alguno de estos tres colores (rojo, verde, azul)' ,'');
  switch (color) {
    case 'rojo': document.write('se ingresó rojo');
                 break;
    case 'verde': document.write('se ingresó verde');
                 break;
    case 'azul': document.write('se ingresó azul');
                 break;
    default:document.write('El color ingresado no es válido');
  }
</script>

</body>
</html>
```

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

```
    case 'rojo': document.write('se ingresó rojo');
                 break;
```

Actividad Nº3:

Solicitar el ingreso alguna de estas palabras (casa, mesa, perro, gato, auto, hombre, mujer) luego mostrar la palabra traducida en inglés. Es decir, si se ingresa 'casa' debemos mostrar el texto 'house' en la página.

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:

Ejemplo07.html

<!DOCTYPE html>

```
<html>
<head>
  <title>Ejemplo de JavaScript</title>
  <meta charset="UTF-8">
</head>
<body>
<script>
  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 etiqueta HTML de **
.**

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.
Es importante decir que NO debe haber un punto y coma al final del while, si hacemos esto estamos en presencia de un error lógico.
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 y los valores que toman las variables a lo largo de la ejecución:

x
1
2
3
4
.
.
100

Cuando x vale 101 la condición de la estructura repetitiva retorna falso, en este caso finaliza la estructura repetitiva.
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).

Actividad Nº4:

Diseñe una página que permita el ingreso de un número entero.
Luego, deberá mostrar la tabla de multiplicar de ese número desde el factor 1 y hasta el 12, por ejemplo si ingreso 5 la salida deberá ser:

5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
.
.
.
.
5 x 12 = 60

Actividad Nº5:

Modifique la actividad anterior de manera que los resultados mostrados efectivamente se muestren en una tabla. La salida debería ser algo similar a:

Número	Operación	Factor	Resultado
5	X	1	5

5	X	2	10
5	X	3	15
5	X	4	20
5	X	5	25
5	X	6	30
5	X	7	35
5	X	8	40
5	X	9	45
5	X	10	50
5	X	11	55
5	x	12	60

Use estilos donde sea necesario.

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:

1 - Primero se ejecuta por única vez el primer argumento <Inicialización>.

Por lo general se inicializa una variable.

2 - El segundo paso es evaluar la (Condición), en caso de ser verdadera se ejecuta

el bloque, en caso contrario continúa el programa.

3 - El tercer paso es la ejecución de las instrucciones.

4 - El cuarto paso es ejecutar el tercer argumento (Incremento o Decremento).

5 - 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.

Ejemplo08.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de JavaScript</title>
  <meta charset="UTF-8">
</head>
<body>

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

Actividad Nº6:

Modifique la actividad nº5 resolviéndola con un lazo **for**.

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:

Ejemplo09.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de JavaScript</title>
  <meta charset="UTF-8">
</head>
<body>


<script>
  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:

Ejemplo10.html

```
<!DOCTYPE html>
<html>
<head>
 <title>Ejemplo de JavaScript</title>
 <meta charset="UTF-8">
</head>
<body>

<script>
 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 mostrarMensaje (es decir la segunda palabra con la primer letra en 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.

Acotaciones
Hemos estado haciendo uso de funciones existentes por defecto en JavaScript como son:
prompt
parseInt
Recordemos que 'prompt' es una función que nos permite ingresar por teclado una cadena de caracteres, y la función 'parseInt' convierte una cadena de caracteres a tipo de dato entero.

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.

Ejemplo11.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de JavaScript</title>
  <meta charset="UTF-8">
</head>
<body>

<script>
  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 : Confeccionar una función que reciba un valor entero comprendido entre 1 y 5. Luego retornar en texto el valor recibido.

Ejemplo12.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de JavaScript</title>
  <meta charset="UTF-8">
</head>
<body>

<script>
  function convertirEnTexto(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=convertirEnTexto(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 : 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 27 de enero de 2020".

Ejemplo13.html

```
<!DOCTYPE html>
<html>
<head>
 <title>Ejemplo de JavaScript</title>
 <meta charset="UTF-8">
</head>
<body>

<script>

 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(27,1,2020));
</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(27,1,2020);
  document.write(fec);
```

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

Actividad Nº 7:

Realice una página con un script. Este debe contener una función **Operaciones** que recibirá como parámetros 2 números y la operación que se desea realizar (suma, resta, multiplicación, división). Con estos datos la función devolverá un string similar a "El resultado de la Operación xxx de los números x e y es igual a zzz"

Por ejemplo si se ingresa 8 y 9 y multiplicar como operación, la función deberá devolver "El Resultado de la Operación Multiplicar de los números 8 y 9 es 72"

Se solicitará el ingreso de los dos números y la operación a realizar.

Actividad Nº 8:

Realice una página html que poseerá un script en el cual habrá una función **calcularPromedio** que recibirá 4 números. Con ellos calculará el promedio y lo mostrará por pantalla.

Actividad Nº 9:

Realice una página html que poseerá un script en el cual habrá una función **generarTabla** que recibirá como parámetros la cantidad de filas y la cantidad de columnas de dicha tabla. Con esos valores generará una tabla html que se mostrará en dicha página.

Actividad Nº 10:

Realice una página html que poseerá un script en el cual habrá una función **funcionesTrigonometricas** que recibirá como parámetro el valor en grados de un ángulo Con esos valores generará una tabla html en que mostrará el seno, coseno y tangente del ángulo ingresado, de su suplementario (90 - ángulo) y de su complementario (180 - ángulo) . El resultado deberá ser similar a:

Función	Angulo	Complementario	Suplementario
	50	40	130
seno	0.76604	0.64278	0.76604
coseno	0.64278	0.76604	-0.64278
tangente	1.19175	0.83909	-1.19175