TEMA 4.

Introducción a la programación con el lenguaje JavaScript

Tecnologías de la Información

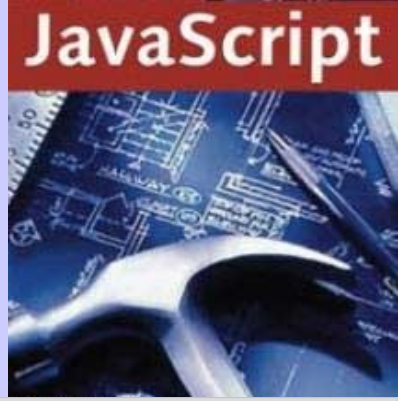
1ºBachillerato

IES Zurbarán (Badajoz)

Sesión 7

Estructuras de control repetitivas: bucles for, while y do..while

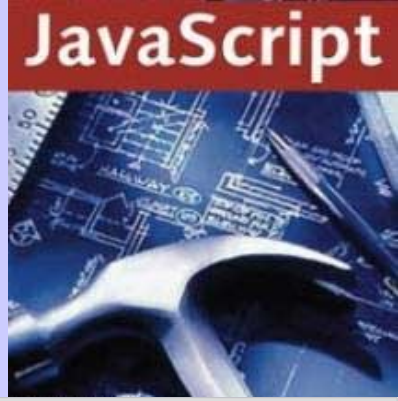
1 ¿Qué es un Bucle?



- Un **bucle** no es más que la "**repetición de una o varias sentencias mientras se cumpla una condición**".
  - *Es una de las estructuras más repetidas en programación.*
- Existen diferentes **tipos** de bucles. Veremos los tres más usados:
  - Bucle **FOR**
  - Bucle **WHILE**
  - Bucle **DO WHILE**
- Cada uno de ellos está indicado para un tipo de iteración(*) distinto, aunque con los tres podemos llegar a conseguir el mismo objetivo.
- La elección de uno u otro depende del programador.

(*) *Acción y efecto de **iterar** (repetir)*

2 Sentencia FOR



- El **bucle FOR** se utiliza para "repetir una o más instrucciones un número fijo de veces".

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

Fijémonos en varias cosas:

Para empezar vemos como unas **llaves** encierran las acciones (sentencias) que queremos realizar mientras se cumpla la condición.

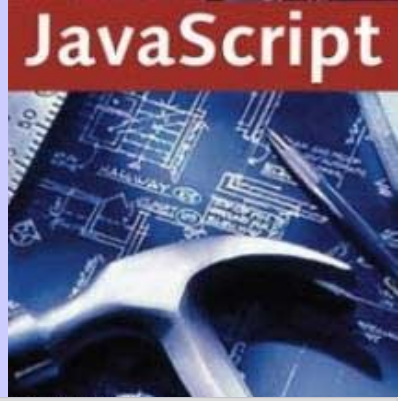
Estas llaves han de colocarse siempre, excepto en el caso de que sólo haya una instrucción como acción a realizar, que son opcionales.

Observa como el **bucle FOR** tiene tres partes encerradas entre paréntesis:

a) La primera es la **inicialización**, que tiene efecto solamente antes de comenzar la primera repetición del bucle. En esta parte se suele colocar la variable que utilizaremos para llevar la cuenta de las veces que se ejecuta el bucle (*variable de control*).

b) La segunda parte es la **condición**, que se evaluará cada vez que comience la repetición del bucle. Contiene una expresión condicional, que de cumplirse, permite la ejecución de las acciones encerradas entre llaves.

c) Por último tenemos la **actualización**, que sirve para indicar los cambios que queramos ejecutar en la variable de control cada vez que termina la repetición del bucle y antes de comprobar si se debe seguir ejecutando.



3 Sentencia FOR

Ejemplo

- Veamos un **ejemplo** que use el **bucle FOR**:

```
var i
for (i=0;i<=10;i++) {
    document.write('Vuelta ' + i)
}
```

- Vamos a ver como sería la ejecución

Repetición	Valor i	¿i<=10?	Salida Pantalla
1	0	SI	Vuelta 0
2	1	SI	Vuelta 1
3	2	SI	Vuelta 2
4	3	SI	Vuelta 3
5	4	SI	Vuelta 4
6	5	SI	Vuelta 5
7	6	SI	Vuelta 6
8	7	SI	Vuelta 7
9	8	SI	Vuelta 8
10	9	SI	Vuelta 9
11	10	SI	Vuelta 10
12	11	NO	**FIN**

Fíjate en la variable **i**.

Mediante esta variable manejamos todo el **bucle FOR**.

El paréntesis nos indica el funcionamiento del bucle:

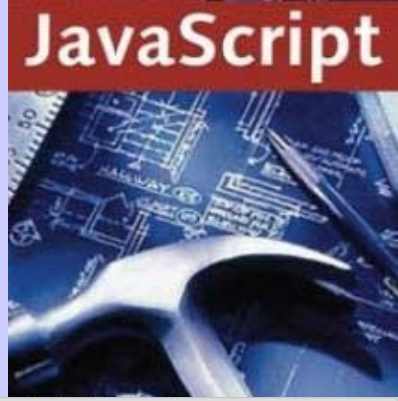
a) i=0: la variable es inicializada a 0 *(valor inicial)*.

b) i<=10. esta es la condición. El bucle se repetirá mientras i sea menor o igual que 10.

c) i++: acción a realizar una vez terminada la iteración y antes de la comprobación de la condición. En este caso, incrementar en 1 la variable de control.

3 Sentencia FOR

Ejemplos



- Tener en cuenta que el **bucle FOR** está dirigido por la variable que aparece en el paréntesis (*en el ejemplo anterior, la variable **i***).
- Para **incrementar** o **decrementar** esta variable, usamos los siguientes operadores:
  - *Autoincremento: **i++*** *Autodecremento: **i--***
  - *Incrementar en varias unidades: **i+=x**, donde x es el número de unidades que incrementamos la variable*
  - *Decrementar en varias unidades: **i-=x**, donde x es el número de unidades que decrementamos la variable*

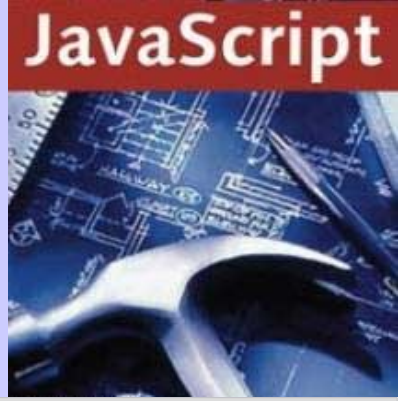
```
for (i=1;i<=1000;i+=2) {
    document.write(i)
}
```

En este ejemplo, escribimos los números impares comprendidos entre el 1 y el 1000. Para ello se incrementa i en dos unidades cada vuelta.

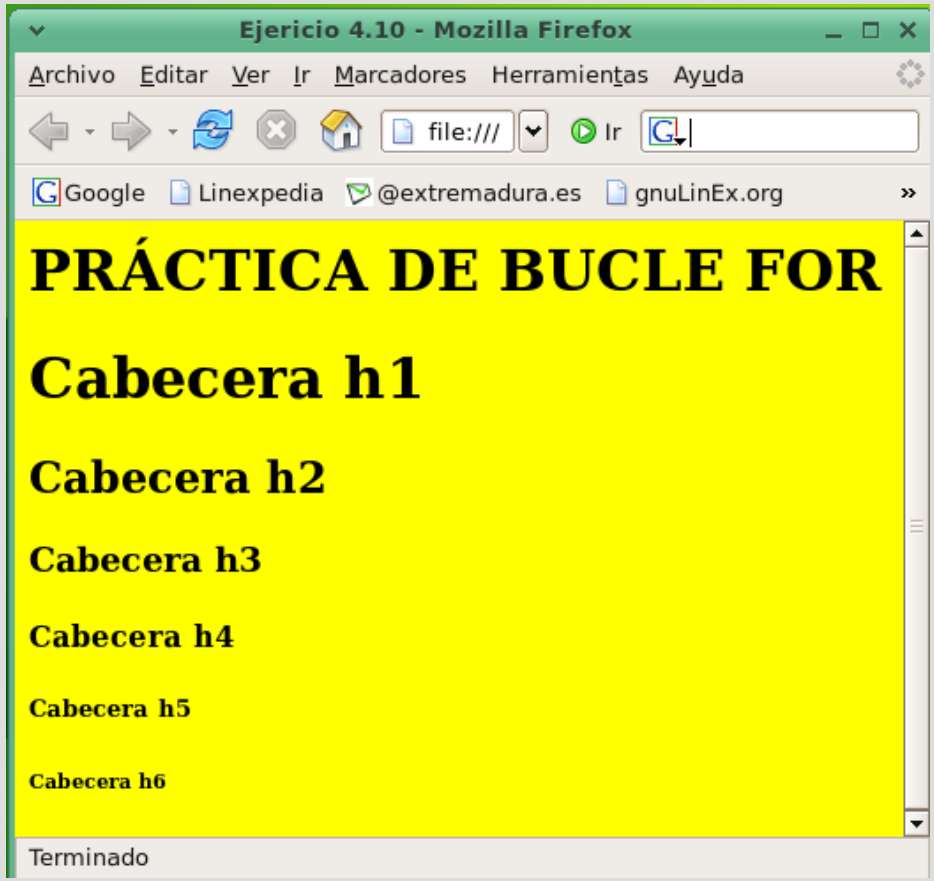
```
for (i=343;i>=10;i--) {
    document.write(i)
}
```

En este otro, escribimos los números enteros comprendidos entre el 343 y el 10 descendentemente.

Ejercicio 4.10.



Escribir un programa que muestre la frase "*Cabecera h*" seguido del número. Las frases deben estar formateadas con las etiquetas adecuadas. El resultado debe ser el siguiente:



Para realizar este ejemplo necesitarás saber como se unen cadenas de caracteres usando **document.write**.

Para hacerlo, basta con utilizar el operador + entre las cadenas de caracteres que queremos unir.

Por ejemplo:

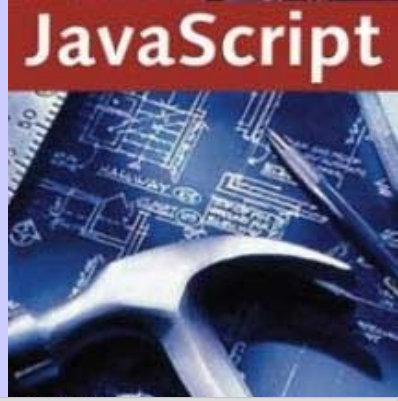
```
var texto;

texto = ' contigo ';
document.write('Sueño ' + texto + ' princesa.')
```

Escribirá en pantalla:

Sueño contigo princesa

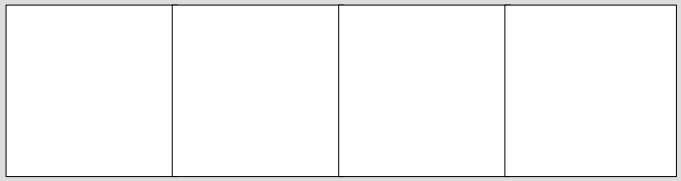
Ejercicio 4.11.



Hacer un programa que pida por teclado tres valores: el número de columnas de una tabla y, la altura y anchura (en pixels) de sus celdas. Una vez tecleados estos valores, el programa pintará en la página web una tabla HTML de una fila por el nº de columnas tecleadas.

Ejemplo:

nº columas --> 4 columnas
alto --> 50 pixels
ancho --> 50 pixels



El código Javascript debe ser capaz de escribir:

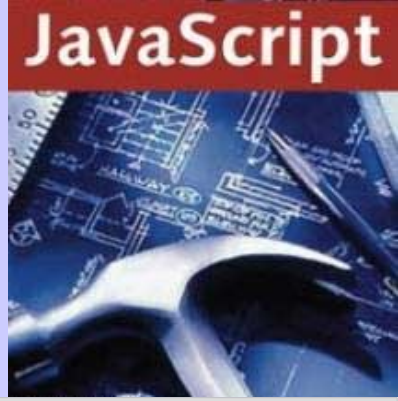
```
<table border="0" cellspacing="2" bgcolor="black" width="200">
  <tr bgcolor="white" height="50">
    <td width="50"> </td>
    <td width="50"> </td>
    <td width="50"> </td>
    <td width="50"> </td>
  </tr>
</table>
```

dos sentencias **document.write** antes del bucle for.

hacer esto con un **bucle for**

otras dos sentencias **document.write** después del bucle for

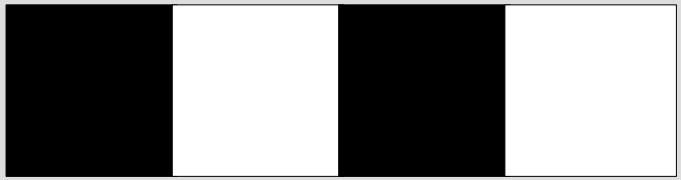
Ejercicio 4.12.



Modifica el ejercicio anterior para que las columnas impares tengan un fondo negro y las pares fondo blanco.

Ejemplo:

nº columas --> 4 columnas
alto --> 50 pixels
ancho --> 50 pixels



El código Javascript debe ser capaz de escribir:

```
<table border="0" cellspacing="2" bgcolor="black" width="200">
  <tr height="50">
    <td width="50" bgcolor="black"> </td>
    <td width="50" bgcolor="white"> </td>
    <td width="50" bgcolor="black"> </td>
    <td width="50" bgcolor="white"> </td>
  </tr>
</table>
```

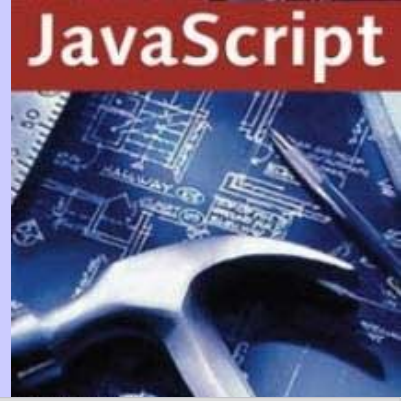
dos sentencias **document.write** antes del bucle for.

hacer esto con un **bucle for**

otras dos sentencias **document.write** después del bucle for

Si el resto de la división de cualquier número entre dos es 1, se dice que el número es **par**. Si da 0, el número es **impar**. En JavaScript se usa el **operador %** para obtener el resto de una división. *Por ejemplo, 3424%2 daría 0, lo que nos dice que el número 3424 es par*.
En el ejercicio debemos dividir la **variable de control** entre 2 y comprobar con un **if** si es igual o no a 1. En un caso escribimos una línea <td ... bgcolor="black"> ... y en el otro con bgcolor="white". Esto se hace dentro del **bucle for**.

4 Bucle WHILE (1/3)



- Estos bucles se utilizan cuando queremos "**repetir la ejecución de unas sentencias un número indefinido de veces, siempre que se cumpla una condición**".
- Es más sencillo de comprender que el **bucle FOR**, pues no incorpora en la misma línea la inicialización de las variables, su condición para seguir ejecutándose y su actualización. Sólo se indica, como veremos a continuación, la condición que se tiene que cumplir para que se realice una iteración.

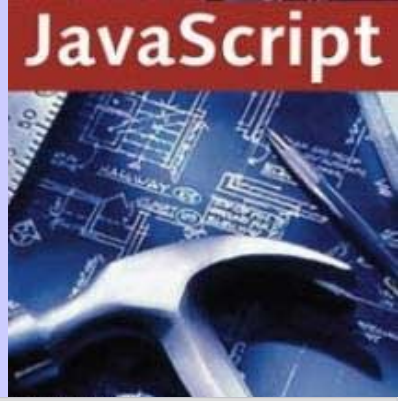
Como vemos, este bucle es idéntico al primer for que estudiamos antes, aunque como podemos apreciar, mucho más simple.

En realidad, la complejidad de for se ha descompuesto en partes separadas sencillas.

```
var i;

i=1;
while (i < 10) {
  document.write('Estoy en la vuelta: ' + i + '<br>');
  i++;
}
```

4 Bucle WHILE (2/3)



- ¿Para qué dos bucles si, al fin y al cabo, hacen lo mismo?
- La respuesta se encuentra en la definición de cada bucle. En el caso del **bucle FOR**, iteramos siempre un número fijo de veces. Esto no ocurre con el **bucle WHILE**, donde el número de iteraciones (*veces que se repiten las intrucciones del cuerpo del bucle*) es indeterminado: depende de las condiciones de ejecución del programa.
- A pesar de todo, podemos hacer que un bucle FOR itere un numero indefinido de veces. Ello se consigue usando una sentencia if dentro del **bucle FOR** con la condición de salida:

Bucle sin inicialización, condición de salida ni actualización. Estamos ante un bucle "infinito", sin salida.

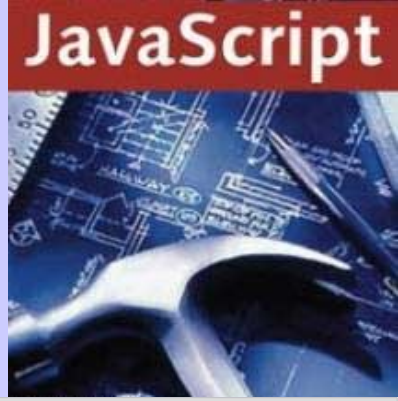
```
var nombre;

for(;;) {
  nombre=prompt('Cómo te llamas? ','');
  if (nombre=='israel') break;
}
```

Saldremos del bucle cuando se cumpla la condición de la sentencia **if**. Salimos ejecutando la instrucción **break**

4 Bucle WHILE (3/3)

Ejemplo




```
var color;


color = '';
while (color != 'rojo') {
  color = prompt('Dame un color')
}
```

En este ejemplo, el programa pide al usuario el nombre de un color.

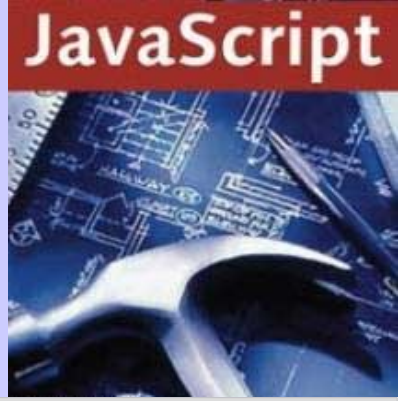
Hasta que el usuario no escriba la palabra **rojo,** el programa no dejará de pedirle nombres de colores.

Para ejecutar un bucle como éste, **primero** tenemos que **inicializar** (*darle un valor inicial para que se pueda realizar la primera comparación*) **la variable** que vamos utilizar en la condición de iteración del bucle.

Con la variable inicializada podemos escribir el bucle. Éste, comprobará si la variable color es distinto de “rojo". En caso aformativo, pedirá otro color y se volverá a repetir el proceso.

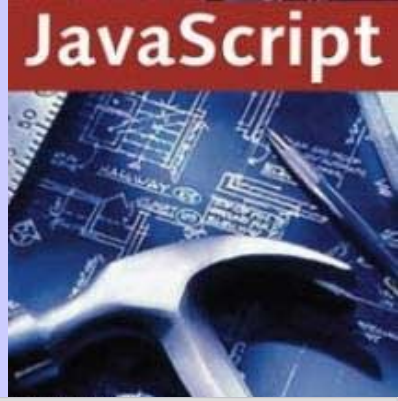
Solamente saldremos del bucle y por tanto del programa, cuando acertemos con la palabra rojo.

Ejercicio 4.13.



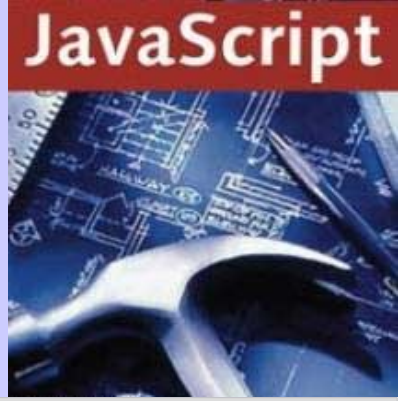
Adapta el ejercicio 4.11 utilizando un **bucle while** en vez de un bucle for.

Ejercicio 4.14.



Adapta el ejercicio 4.12 utilizando un **bucle while** en vez de un bucle for.

Ejercicio 4.15.



Escribe un programa en JavaScript que consista en adivinar un número previamente introducido por teclado (jugador 1).

El programa pedirá tantos números como intentos erróneos haga el jugador 2. Solamente terminará cuando el jugador 2 acierte.

El programa dará pistas al jugador 2 indicándole si su número es mayor o menor que el número a adivinar.

Usa dos variables: num_adivinar y num.

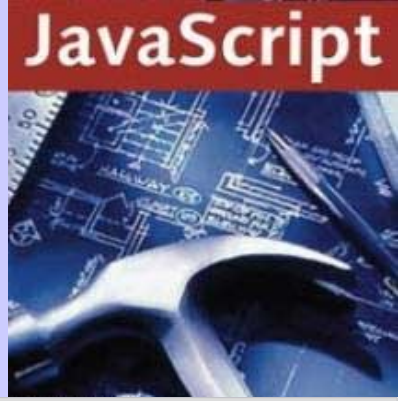
Necesitarás un **bucle while.**

Piénsa en la expresión condicional de deberás usar en el bucle.

Además necesitas una sentencia **if** para decirle al jugador 2 si su número es mayor o menor que el número a adivinar.

Usar la sentencia **alert** para mostrar un mensaje al usuario por pantalla. Ejemplo: **alert**('El número es mayor');

4 Bucles DO..WHILE



- Se utiliza generalmente cuando no se sabe cuantas vueltas dará el bucle.
- Es parecido al bucle **while** pero con la sutil diferencia de que sabemos seguro que "**el bucle por lo menos se ejecutará una vez**".
- Cualquier problema escrito con do**...while** se puede escribir también utilizando un bucle **while**.
- Sintaxis:

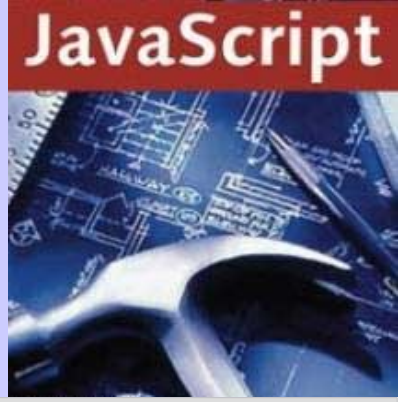
```
do {
    sentencias del bucle
} while (condición)
```

El bucle se ejecuta, al menos, una vez.

Una vez ejecutadas las sentencias encerradas entre llaves, se evalúa la condición.

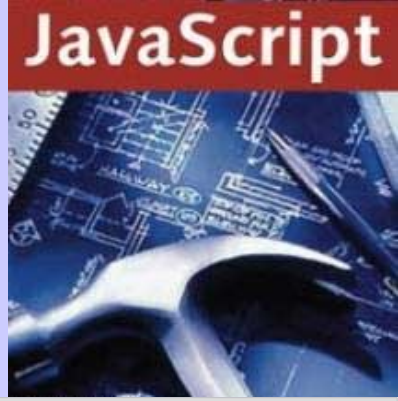
Si se cumple, se vuelve a dar otra pasada. En caso contrario se sale del bucle..

Ejercicio 4.16.



Adaptar el ejercicio 4.15. al bucle DO..WHILE

5 Bucles anidados



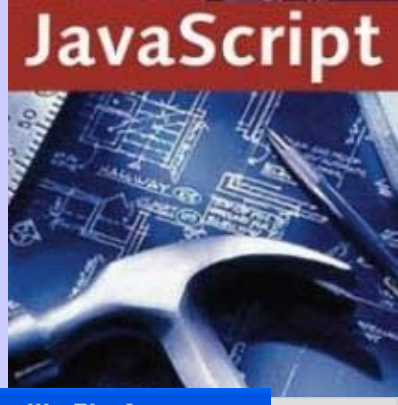
- Anidar un bucle consiste en "**meter ese bucle dentro de otro**".
- La anidación de bucles es necesaria para hacer determinados procesamientos un poco más complejos que los que hemos visto en los ejemplos y ejercicios anteriores.
- Un bucle anidado tiene una estructura como la que sigue:

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

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

El bucle que está anidado (*más hacia dentro*) es el que más veces se ejecuta, en este ejemplo, para cada iteración del bucle más externo el bucle anidado se ejecutará por completo una vez, es decir, hará sus 10 iteraciones. En la página web se escribirían estos valores, en la primera iteración del bucle externo y desde el principio: **0-0 0-1 0-2 0-3 0-4 0-5 0-6 0-7 0-8 0-9**

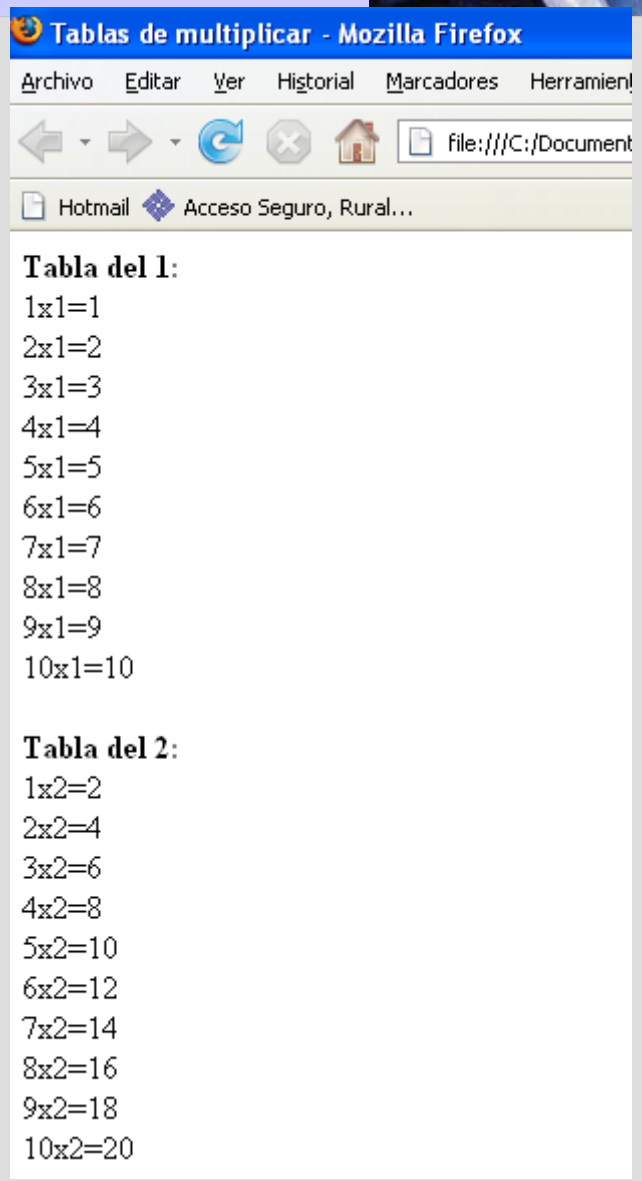
Para cada iteración del bucle externo se ejecutarán las 10 iteraciones del bucle interno o anidado. Hemos visto la primera iteración, ahora vamos a ver las siguientes iteraciones del bucle externo. En cada una acumula una unidad en la variable i, con lo que saldrían estos valores. **1-0 1-1 1-2 1-3 1-4 1-5 1-6 1-7 1-8 1-9** y así sucesivamente hasta tomar i el valor 9: **9-0 9-1 9-2 9-3 9-4 9-5 9-6 9-7 9-8 9-9**



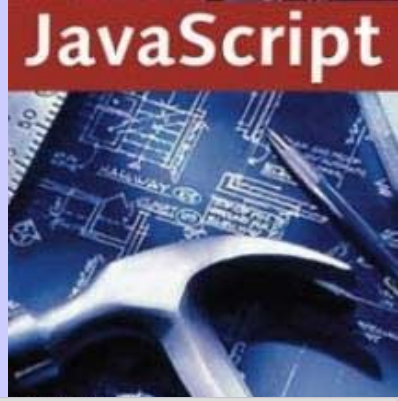
Ejercicio 4.17.

Hacer un programa en JavaScript que usando dos bucles anidados for imprima por pantalla todas las tablas de multiplicar.

El resultado debería ser algo parecido a:



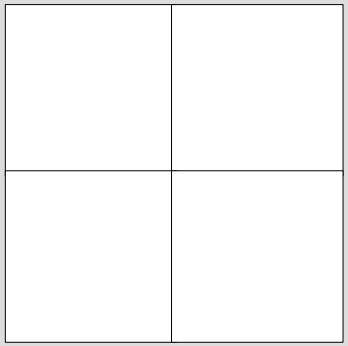
Ejercicio 4.18.



Hacer un programa que pida por teclado cuatro valores: el número de columnas y filas de una tabla y, la altura y anchura (en pixels) de sus celdas. Una vez tecleados estos valores, el programa pintará en la página web una tabla HTML del nº de filas por el nº de columnas tecleadas.

Ejemplo:

nºcolumas --> 2 columnas
nºfilas --> 2 filas
alto --> 50 pixels
ancho --> 50 pixels



El código Javascript debe ser capaz de escribir:

```
<table border="0" cellspacing="2" bgcolor="black" width="200">
  <tr bgcolor="white" height="50">
    <td width="50"> </td>
    <td width="50"> </td>
  </tr>
  <tr bgcolor="white" height="50">
    <td width="50"> </td>
    <td width="50"> </td>
  </tr>
</table>
```

} una sentencia **document.write** antes del bucle for más externo.

} hacer esto con el **bucle for** interno (para i=1)

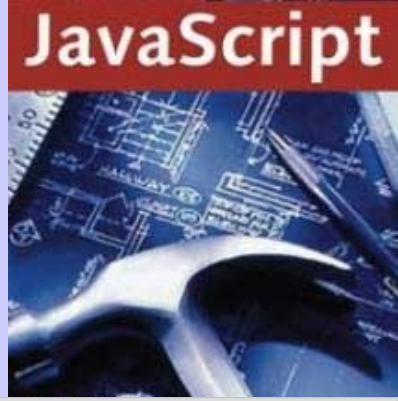
} hacer esto con el **bucle for** externo (para j=1)

} hacer esto con el **bucle for** interno (para i=1)

} hacer esto con el **bucle for** externo (para j=2)

} otra sentencia **document.write** después de los dos bucles for

Ejercicio 4.19.



Hacer un programa que pinte un **tablero de ajedrez** de 8x8 casillas usando dos for anidados.

El programa pedirá solamente el ancho de la celda que será igual que el alto.

Ejemplo:

ancho y ancho --> 50 pixels