
DESARROLLO WEB EN ENTORNO CLIENTE
TEMA 2:
Introducción al lenguaje JavaScript
JS

Características de JavaScript

- ¿Qué es JavaScript?

Lenguaje de programación interpretado utilizado fundamentalmente para dotar de comportamiento dinámico a las páginas web.

Cualquier navegador web actual incorpora un intérprete para código JavaScript.

Características de JavaScript

- Su sintaxis se asemeja a la de C++ y Java.
- Es un lenguaje débilmente tipado.(una variable puede contener distintos valores)
- La declaración es opcional
- Maneja objetos dentro de nuestra página Web y sobre ese objeto podemos definir diferentes eventos. Dichos objetos facilitan la programación de páginas interactivas.
- Es dinámico, responde a eventos en tiempo real.
- Funcionalidades y limitaciones.pdf

Incorporando JavaScript

1-JavaScript en el mismo documento HTML.

- Uso de unas etiquetas predefinidas para marcar el texto **(<script> y </script>).**
- Puede incluirse en cualquier parte del documento, aunque se recomienda que se defina dentro de **la cabecera** del documento HTML.
- Esta técnica suele utilizarse cuando se definen instrucciones que se referenciarán desde cualquier parte del documento o cuando se definen funciones con fragmentos de código genéricos.

Incorporando JavaScript

JavaScript en el mismo documento HTML.

```
<head>
------------
<title>Ejemplo 1</title>
      <script type="text/javascript">
      alert("Prueba de JavaScript");
      </script>
</head>
<body>
<h1>Ejemplo 1: código embebido</h1>
</body>
</html>
```

Incorporando JavaScript

JavaScript en un archivo externo.

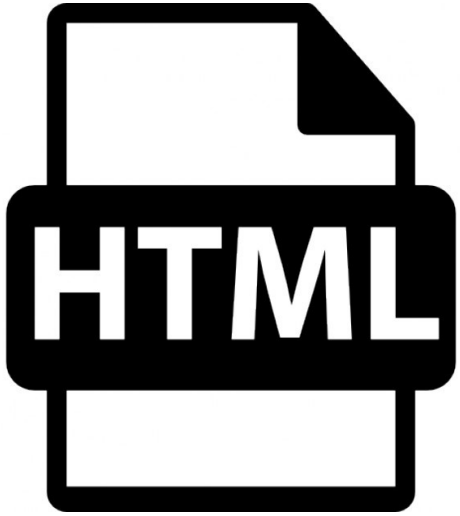
- Las mismas instrucciones de JavaScript que se incluyen entre un bloque **<script></script>** pueden almacenarse en un fichero externo con **extensión .js**.
- La forma de acceder y enlazar esos ficheros .js con el documento HTML/XHTML es a través de la propia etiqueta **<script> o link.**
- No existe un límite en el número de ficheros .js que pueden enlazarse en un mismo documento HTML/XHTML.

Incorporando JavaScript

JavaScript en un archivo externo.

Archivo mensaje.js:

alert("Prueba de JavaScript");



- <head>
- ----------
- <title>Ejemplo 2</title>
- **<script type="text/javascript" src="/inc/mensaje.js"></script>**
- </head>
- <body>
- <h1>Ejemplo 2: fichero externo</h1>
- </body>
- </html>

Incorporando JavaScript

JavaScript en elementos HTML.

- Consiste en insertar fragmentos de JavaScript **dentro de atributos de etiquetas HTML** de la página.
- Forma de **controlar los eventos** que suceden asociados a un elemento HTML concreto.
- Principal **desventaja**: el mantenimiento y modificación del código puede resultar más complicado.

Incorporando JavaScript

JavaScript en elementos HTML.

- <head>

- <title>Ejemplo 3</title>
- </head>
- <body>
- <p onclick="**alert(‘Prueba de JavaScript’);”>**
- </p>
- </body>
- </html>

Incorporando JavaScript

Se utiliza para definir un contenido alternativo en el documento web, cuando el navegador no soporta la ejecución de scripts, o bien cuando los mismos están desactivados en el navegador que recibe el documento.

<NOSCRIPT>
Contenido a visualizar o cargar por el navegador
< /NOSCRIPT>

Incorporando JavaScript

- La etiqueta se puede utilizar dentro del encabezado del documento, suele afectar a etiquetas como <link>, <style> y <meta>, de forma que permite realizar la carga alternativa de estos elementos en caso de que el navegador no permite la ejecución de scripts.

```
<!DOCTYPE HTML>
<html>
<head>
<title>Ejemplo etiqueta noscript</title>
<meta charset="utf-8" >
<meta name="author" content="Daw2"/>
<meta name="description" content="Ejercicios HTML5"/>
<link rel="stylesheet" type="text/css" href="styles.css"/>
<link rel="shortcut icon" href="favicon.ico" type="image/x-icon" />
<script src="simple2.js"></script>
<noscript>
<h1>Se requiere Javascript</h1>
<p>No se puede usar esta pagina sin Javascript</p>
</noscript>
</head>
<body>
<p>
Ejercicios y practicas de programación Web HTML5
</p>
<a href="http://www.html5aprendo.com">Visite aprendo.com</a>
<a href="page2.html">Pagina 2</a>
</body>
</html>
```


Ejemplo etiqueta noscript
Configuración
localhost/ejercicioshtml5/index1.html
CAM.ES : Inicio
Gmail: el correo elec...
Otros marcadores
Se requiere Javascript
No se puede usar esta pagina sin Javascript
Ejercicios y practicas de programación Web HTML5
Visite aprendoencasa.com
Pagina 2

El siguiente ejemplo muestra el uso de la etiqueta dentro de la sección de encabezado, realizado una redirección a otro documento web en caso de no poder ejecutarse los scripts.

```
<noscript>
<meta http-equiv="refresh" content="0;
   http://www.redirección “/>
</noscript>
```

Comprobar deshabilitando javascript en el navegador.

El lenguaje Js: Sintáxis

Mayúsculas y minúsculas:

- El lenguaje distingue entre mayúsculas y minúsculas, a diferencia de por ejemplo HTML.
- No es lo mismo utilizar alert() que Alert()

El lenguaje Js: Sintáxis

Comentarios en el código:

- Los comentarios no se interpretan por el navegador.
- Existen dos formas de insertar comentarios:
  - Doble barra (//) –Se comenta una sola línea de código.
  - Barra y asterisco (/* al inicio y */ al final) –Se comentan varias líneas de código.

El lenguaje Js: Sintáxis

Comentarios en el código –Ejemplo:

- <script type="text/javascript">
- // Este modo permite comentar una sola línea
- /* Este modo permite realizar
- comentarios de
- varias líneas */
- </script>

El lenguaje Js: Sintáxis

Tabulación y saltos de línea:

- JavaScript ignora los espacios, las tabulaciones y los saltos de línea con algunas excepciones.
- Emplear la tabulación y los saltos de línea mejora la presentación y la legibilidad del código.

El lenguaje Js: Sintáxis

```
<script type="text/javascript">
  var i,j=0;
  for (i=0;i<5;i++){
    alert("Variable i: "+i;
    for (j=0;j<5;j++){
      if (i%2==0){
        document.write(i + "-" + j + "<br>");
      }
    }
  }
</script>
```

El lenguaje Js: Sintáxis

El punto y coma:

- Se suele insertar un signo de punto y coma (;) al final de cada instrucción de JavaScript.
- Su utilidad es separar y diferenciar cada instrucción.
- Se puede omitir si cada instrucción se encuentra en una línea independiente (la omisión del punto y coma no es una buena práctica de programación).

El lenguaje Js: Sintáxis

Palabras reservadas:

- Algunas palabras no se pueden utilizar para definir nombres de variables, funciones o etiquetas.
- Es aconsejable no utilizar tampoco las palabras reservadas para futuras versiones de JavaScript.

En http://msdn.microsoft.com/es-es/library/ie/0779sbks(v=vs.94).aspx

es posible consultar todas las palabras reservadas de JavaScript.

Escritura de cadenas de texto en la página

- document.write("Texto")
- document.write('Texto')
- document.write('<IMG SRC="imagen.gif">')

Cuadros de diálogo

La manera más sencilla consiste en utilizar el método **alert()** del objeto **window**:

- alert("Texto")

Por ejemplo:

- alert("¡Bienvenido!\n\tEsta Web está dedicada a JavaScript.")
- Este tipo de ventanas de diálogo muestra un botón de aceptar, normalmente con el nombre *Aceptar*

Cuadros de diálogo

El método **prompt()** del objeto **window** muestra una caja de texto en la que el usuario puede introducir contenidos. También muestra dos botones, *Aceptar* y *Cancelar*. Ejemplo:

- prompt("Su color favorito es: ","Azul")
- El método **confirm()** del objeto window nos devuelve true o false dependiendo del botón pulsado.

Cuadros de diálogo

```
<HTML>
    <HEAD>
    <TITLE>Ejemplo : página que pide el nombre</TITLE>
    </HEAD>
    <BODY>
    <SCRIPT LANGUAGE="JavaScript">

    var nombre = prompt("Introduzca su nombre:","")
    document.write("<H2>Bienvenido, " + nombre + "</H2>")

    </SCRIPT>
    <P>Aquí va el resto de la página...</P>
    </BODY>
    </HTML>
```

Tipos de datos

Tipos de datos

- Los tipos de datos especifican qué tipo de valor se guardará en una determinada variable.(tipo dinámico)
- Los tipos de datos nativos de JavaScript son:
  - Numérico.
  - Cadena.
  - booleano.
  - -undefined
  - -null
  - -objeto

Tipos de datos

Números:

- En JavaScript existe sólo un tipo de dato numérico.

- JavaScript permite representar cualquier número que sea necesario. Ten en cuenta que esto no significa que se puedan usar números “descomunales” porque un computador siempre ha de tener un límite. Pero el límite de JavaScript es del orden de $\pm 1.7976931348623157 \times 10^{308}$

- Para indicar que un número es negativo se precede del signo menos.

- Los números decimales se escriben utilizando el punto (.) como separador. Si la parte entera de un número es el cero, se admite omitir el cero. Es decir, 0.55 y .55 son ambos admitidos.
- También se admite la notación basada en indicar un número seguido de E y la potencia de 10 a la que se debe elevar.

- Todos los números se representan a través del formato de punto flotante de 64 bits.
- Este formato es el llamado double en los lenguajes Java o C++.

- **Valores especiales para variables numéricas**
- **NaN: no es un número.**
- **Infinity: infinito, por ejemplo 3/0**

Tipos de datos

Cadena:

- El tipo de datos para representar cadenas de texto se llama string.
- Se pueden representar letras, dígitos, signos de puntuación o cualquier otro carácter de Unicode.
- La cadena de caracteres se debe definir entre comillas dobles o comillas simples.

Tipos de datos

- Cadenas de texto -Secuencias de escape:

Secuencia de escape	Resultado
\\	Barra invertida
\'	Comilla simple
\"	Comillas dobles
\n	Salto de línea
\t	Tabulación horizontal
\v	Tabulación vertical
\f	Salto de página
\r	Retorno de carro
\b	Retroceso

Tipos de datos

booleano:

- También conocido como valor lógico.
- Sólo admite dos valores: true o false.
- Es muy útil a la hora de evaluar expresiones lógicas o verificar condiciones.

Tipos de datos

Null:

- Se utiliza para inicializar variables que van a contener objetos.
- Null significa que el objeto no ha sido creado.

Tipos de datos

- Objeto

Js esta orientado a objeto pero no trabaja con clases.

Para poder crear un objeto siempre tendrá que r acompañado de new.

Antes de crearle

Var obj=new Object();

Tipos de datos

- Undefined
- Variables no inicializadas

Identificación de tipos

- anexo

Variables

- Se pueden definir como zonas de la memoria de un ordenador que se identifican con un nombre y en las cuales se almacenan ciertos datos.
- Si una variable no contiene ningun valor su contenido es **null**
- El desarrollo de un script conlleva:
  - Declaración de variables.
  - Inicialización de variables.

Variables

Declaración de variables:

- Se declaran mediante la palabra clave var seguida por el nombre que se quiera dar a la variable.

```
Var mi_variable;
```

- Es posible declarar más de una variable en una sola línea.

```
Var mi_variable1, mi_variable2;
```

Podemos declarar variables con **let** en lugar de var si no queremos que sean accesibles más allá de un ámbito

Ejemplo let -var

```
//ES5
(function() {
  console.log(x); // x no está definida aún.
    if(true) {
      var x = "hola mundo";
    }
    console.log(x);
    // Imprime "hola mundo", porque "var" hace que sea global
    // a la función;
})();
```

```
//ES6
(function() {
  if(true) {
      let x = "hola mundo";
    }
    console.log(x);
    //Da error, porque "x" ha sido definida dentro del "if"
})();
```

Variables

- Inicialización de variables:
- Se puede asignar un valor a una variable de tres formas:
  - Asignación directa de un valor concreto.
  - Asignación indirecta a través de un cálculo en el que se implican a otras variables o constantes.
  - Asignación a través de la solicitud del valor al usuario del programa.

Ejemplos:

```
Var mi_variable_1 = 30;
Var mi_variable_2 = mi_variable_1 + 10;
Var mi_variable_3 = prompt(„Introduce un valor:“)
```

Variables

- Aunque en JavaScript no es estrictamente necesario declarar una variable antes de utilizarla, la declaración mediante el uso de la palabras reservadas son **var y let** es siempre recomendable, por claridad.
- El nombre de una variable puede empezar por una letra o por el carácter _. Después ya pueden ponerse números. Los nombres de variables son **sensibles a mayúsculas y minúsculas.**

Tipo especial

- Un tipo especial de variable es la constante.
- Con las mismas restricciones sintácticas que las otras variables, pero en estos tipo hay imposibilidad de cambiar su valor una vez establecido.
- Palabra identificadora **const (obligatoria)**

```
const  iva=21;
```

Ambito de ejecución

- El ámbito de una variable es aquella zona del código donde la variable puede ser accesible o utilizada. Dependiendo de cómo y donde la declaremos hay tres ámbitos.
- Global
- Local
- Bloque

Ámbito Global

- Puede ser utilizada en cualquier parte del código.
- Se declara fuera de toda función , normalmente al inicio del código.
- Se declaran con var

Ambito local

- Son variable que se declaran dentro de funciones y su ámbito es siempre dentro de la función donde se declaró.
- Se suelen declarar con let.

Ambito Bloque

- La variable se define dentro de lo que llamamos bloque de código: foreach,for,while…(let)
- Solo podremos usarla dentro de ese bloque.

```
//ES5
(function() {
  console.log(x); // x no está definida aún.
    if(true) {
      var x = "hola mundo";
    }
    console.log(x);
    // Imprime "hola mundo", porque "var" hace que sea global
    // a la función;
})();

//ES6
(function() {
  if(true) {
      let x = "hola mundo";
    }
    console.log(x);
    //Da error, porque "x" ha sido definida dentro del "if"
})();
```

Operadores

JavaScript utiliza principalmente cinco tipo de operadores:

- Aritméticos.
- Lógicos.
- De asignación.
- De comparación.
- Condicionales.

Operadores

Operadores aritméticos:

- Permiten realizar cálculos elementales entre variables numéricas.

Operador	Nombre
+	Suma
−	Resta
*	Multiplicación
/	División
%	Módulo
++	Incremento
−−	Decremento

Operadores

Operadores lógicos:

- Combinan diferentes expresiones lógicas con el fin de evaluar si el resultado de dicha combinación es verdadero o falso.

Operador	Nombre
&&	Y
||	O
!	No

Operadores

Operadores de asignación:

- Permiten obtener métodos abreviados para evitar escribir dos veces la variable que se encuentra a la izquierda del operador.

Operador	Nombre
+=	Suma y asigna
-=	Resta y asigna
*=	Multiplica y asigna
/*	Divide y asigna
%=	Módulo y asigna

Operadores

Operadores de comparación:

- Permiten comparar todo tipo de variables y devuelve un valor booleano.

Operador	Nombre
<	Menor que
<=	Menor o igual que
==	Igual
>	Mayor que
>=	Mayor o igual que
!=	Diferente
===	Estrictamente igual
!==	Estrictamente diferente

Operadores

Operadores condicionales:

- Permite indicar al navegador que ejecute una acción en concreto después de evaluar una expresión.

Operador	Nombre
? :	Condicional

expresiónConValorBooleano ? expresión1 : expresión2;

Interpretación: si la expresiónConValorBooleano es cierta se ejecuta la expresión1, y en caso contrario se ejecuta la expresión2.

```
var estado = (edad >= 18) ? "adulto" : "menor";
```

CONVERSION

- **Conversión explícita de tipos**
- **parseFloat(cadena)**
- Toma la "cadena" y la transforma en un número en coma flotante, si es posible.
- parseFloat ("123.456") = 123.456
- parseFloat ("123ABC") = 123
- parseFloat ("ABC") = NaN

CONVERSION

parseInt(cadena, número)

- Devuelve números enteros, el segundo argumento nos permite escoger la base de numeración (entre 2 y 36)
- parseInt ("ABC",16) = 2748 ABC16 = 274810
- Si no especificamos el segundo argumento, por defecto es 10.
- Si la cadena empieza por **0x y no existe el segundo argumento, se entiende que es 16.**
- Si la cadena empieza por 0 y no existe el segundo argumento, se entiende que es 8

toString(argumento)

- Si argumento = número
- Devuelve una cadena que contiene el número
- Puede haber un argumento opcional:
- (13).toString(16) =”d” siendo 1310 = d16
- (13).toString(2) = “1101” siendo 1310 = 11012
- Funciones de conversión

Sentencias Condicionales

- Con las sentencias condicionales se puede gestionar la toma de decisiones y el posterior resultado por parte del navegador.
- Dichas sentencias evalúan condiciones y ejecutan ciertas instrucciones en base al resultado de la condición.
- Las sentencias condicionales en JavaScript son:
  - if.
  - switch.
  - while.
  - for.

Sentencias Condicionales

- if–sintaxis (1):

```
If (expresión){
instrucciones
}
```

If

– Ejemplo de uso

IF.html

Code | Split | Design | Title: Introduccion JavaScript | Check Page

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml">
3  <head>
4  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5  <title>Introduccion JavaScript</title>
6  </head>
7  <body>
8  <script language="javascript">
9
10     var respuesta=prompt("Introduce una palabra","");
11
12         if (respuesta)
13         {
14                 alert(respuesta);
15
16         }
17
18 </script>
19 </body>
20 </html>
```

Sentencias Condicionales

- if–sintaxis (2):

```
If (expresión){
    instrucciones_si_true
} else {
   instrucciones_si_false
}
```

If

– Ejemplo de uso

```
var edad = Number(prompt("Introduce tu edad"));

if (edad > 20) {
  console.log("Tienes más de 20 años");
}
```

```
var edad = prompt("Introduce tu edad");

if (edad > 20) {
  console.log("Tienes más de 20 años");
} else {
  console.log("Tienes 20 años o menos");
}
```

```
var edad = prompt("Introduce tu edad");

if (edad >= 0 && edad < 20) {
  console.log("Tienes menos de 20 años");
} else if (edad >= 20 && edad <= 40) {
  console.log("Tienes entre 20 y 40 años");
} else {
  console.log("Tienes más de 40 años");
}
```

Sentencias Condicionales

- switch–sintaxis:

```
switch(expresión){
case valor1:
        instrucciones a ejecutar si expresión = valor1
break;
case valor2:
        instrucciones a ejecutar si expresión = valor2
break;
case valor3:
        instrucciones a ejecutar si expresión = valor3
break
default:
    instrucciones a ejecutar si expresión es diferente a
    los valores anteriores
}
```

Switch

- Ejemplo de uso

```
switch (estacion) {

  case "primavera":
    // bloque de código
    break;

  case "verano":
    // bloque de código
    break;

  case "otoño":
    // bloque de código
    break;

  case "invierno":
    // bloque de código
    break;

  default:
    // bloque de código si no coincide ningún caso anterior

}
```

Sentencias Condicionales

•while–sintaxis:

(1)

while(expresión){

instrucciones

}

(2)

do{

instrucciones

} while(expresión)

```
1  n = 0;
2  x = 0;
3  while (n < 3) {
4    n ++;
5    x += n;
6  }
```

```
1  do {
2     i += 1;
3     document.write(i);
4  } while (i < 5);
```

Ejercicio

- Script que repite un texto cualquiera, el número de veces que queramos, utilizando

“while”

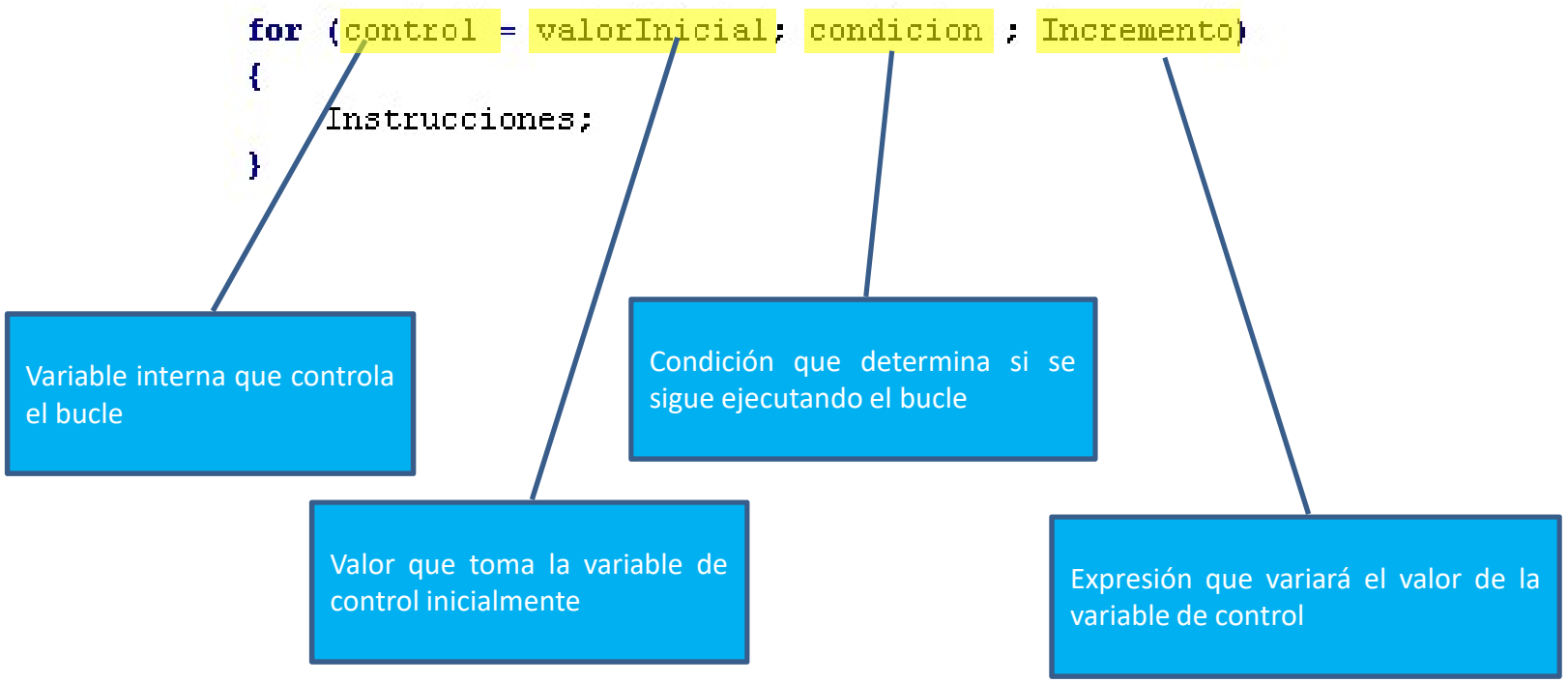
Sentencias Condicionales

- for–sintaxis:

```
for(valor_inicial_variable;
   expresión_condicional;
incremento_o_decremento_de_la_variable){
      cuerpo_del_bucle
}
```

For

- Estructura repetitiva por excelencia



Ejemplos

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

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

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

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

Si queremos contar descendentemente del 343 al 10 utilizaríamos este bucle.

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

Ejercicios de Repaso

```
addEventListener('load',inicio,false);


  function inicio()
  {
Instrucciones
Funciones….
}
```