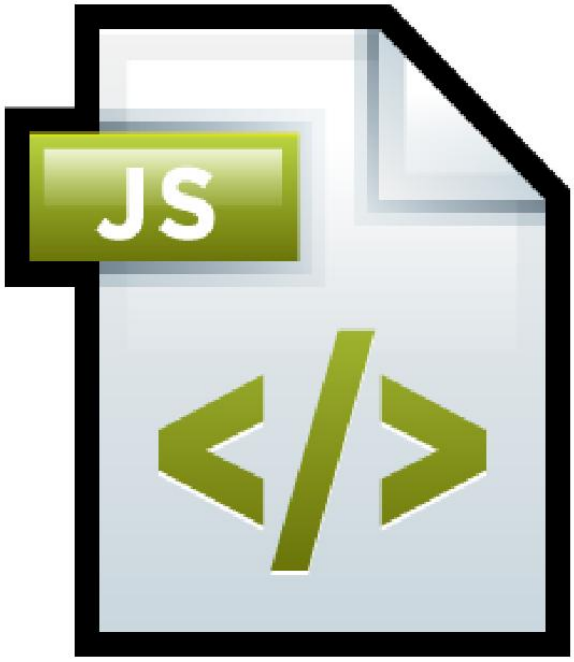


JavaScript. Nivel I

Febrero, 2018

CADIF1
ACADEMIA DE SOFTWARE

Objetivos del nivel

- Conocer las características fundamentales de JavaScript
- Manipular el DOM: Window, Document, Form, etc.
- Validaciones de formularios del lado del cliente

Prerrequisitos del nivel

- HTML 5 y CSS 3 Nivel I
- Lógica de Programación Nivel II

Acerca de este manual

Este manual pertenece al Centro de Asesoramiento y Desarrollo Informático C.A. (CADI F1). Para obtener más información sobre este u otros cursos visite nuestra sitio Web www.cadif1.com, escribanos a la dirección de correo cadi@cadif1.com o visítenos en nuestra sede ubicada en la Av. Pedro León Torres con calle 59, Centro Comercial Sotavento, piso 2 oficina 27, Barquisimeto estado Lara, Venezuela. Tlf. 0251-7179247, 0251-4410268.

Las marcas mencionadas en este manual son propiedad de sus respectivos dueños. Copyright 2018. Todos los derechos reservados.

ACADEMIA DE SOFTWARE

CADI
F1
ACADEMIA DE SOFTWARE

Contenido del nivel

Capítulo 1. Introducción

1.1.- Que es Javascript.
1.2.- Ejecución de Los Scripts.
1.3.- Detección de Errores.

Capítulo 2. Programación Básica

2.1.- Ubicación Del Código Java Scripts.
2.2.- Mensajes Emergentes.
2.3.- Archivos Externos de Javascript.

Capítulo 3. Variables y Tipos de Datos

3.1.- Manejo de Variables.
3.2.- Operadores.
3.3.- La Ventana Prompt.

Capítulo 4. Manipulación de Strings

4.1.- La Clase Strings.
4.2.- Métodos de la Clase String.
4.3.- Conversión de String.

Capítulo 5. Instrucciones Condicionales

5.1.- Condicionales.
5.2.- Confirm.
5.3.- Funciones Especiales.

Capítulo 6. Ciclos

6.1.- For.
6.2.- While.
6.3.- Do While.

ACADEMIA DE SOFTWARE

Capítulo 7. Funciones

7.1.- Definición de Funciones.
7.2.- Parámetros.

Capítulo 8. Eventos

8.1.- Manejadores de Eventos.
8.2.- Eventos y Funciones.

Capítulo 9. Dom - Objeto Window

9.1.- ¿Qué es la jerarquía de objetos?.
9.2.- El Objeto Window.
9.3.- El Método Open.

Capítulo 10. Dom - Document

10.1.- El Objeto Document.
10.2.- Buscar Elementos.
10.3.- Manipular Elementos.

Capítulo 11. Dom - Form

11.1.- El Objeto Form.
11.2.- El Objeto Input.

Capítulo 12. Validación de Formularios. Parte 1

12.1.- Validación de Formularios.
12.2.- Validar Cuadros de Texto.
12.3.- Validar Select.

Capítulo 13. Validación de Formularios. Parte 2

13.1.- Validar correo.
13.2.- Validar teclas.
13.3.- Validar checkbox.

ACADEMIA DE SOFTWARE

Capítulo 14. Manejo de Fechas

14.1.- Date.
14.2.- Settimeout.
14.3.- Setinterval.

CADIF1
ACADEMIA DE SOFTWARE

Capítulo 1. INTRODUCCIÓN

1.1.- Que es Javascript

JavaScript (abreviado comúnmente "JS") es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Javascript es un lenguaje de programación utilizado para crear pequeños programas encargados de realizar acciones dentro del ámbito de una página web. Con Javascript se pueden crear efectos especiales en las páginas y definir interactividades con el usuario.

El navegador es el encargado de interpretar las instrucciones Javascript y ejecutarlas para realizar estos efectos e interactividades, por lo tanto, lo único que se necesita para programar con JavaScript es un navegador. Debido a que JavaScript se ejecuta en entornos muy variados, una parte importante de las pruebas y la depuración es probar y verificar que el código JavaScript funciona correctamente en múltiples navegadores

Entre las acciones típicas que se pueden realizar en Javascript tenemos dos vertientes:

ACADEMIA DE SOFTWARE

- Por un lado los efectos especiales sobre páginas web, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien de color o cualquier otro dinamismo.

- Por el otro, javascript nos permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo que podemos crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo.

Javascript es un lenguaje de programación muy parecido a Java, y por ende muy parecido a lenguaje C, es por esto que tiene casi las mismas características y reglas, entre las cuales destacan:

- El código java script es sensitivo a mayúsculas y minúsculas, es decir, si se define un identificado con el nombre “Edad” es diferente a otro con el nombre “edad”.
- Todas las instrucciones deben terminar con un “;” (excepto algunas pocas)
- Los bloques de código se encierran entre las llaves “{“ y “}”, que indican inicio y fin respectivamente.
- Existen dos tipos de comentarios: comentarios de línea (//) y comentarios de bloque (/* */)

1.2.- Ejecución de Los Scripts

Existen dos maneras de ejecutar scripts en la página. La primera de estas maneras se trata de ejecución directa de scripts, la segunda es una ejecución como respuesta a la acción de un usuario. Veremos ahora cada una de ellas.

a. Ejecución directa
Es el método de ejecutar scripts más básico. En este caso se incluyen las instrucciones dentro de la etiqueta <SCRIPT>, tal como hemos comentado anteriormente. Cuando el navegador lee la página y encuentra un script va interpretando las líneas de código y las va ejecutando una después de otra.

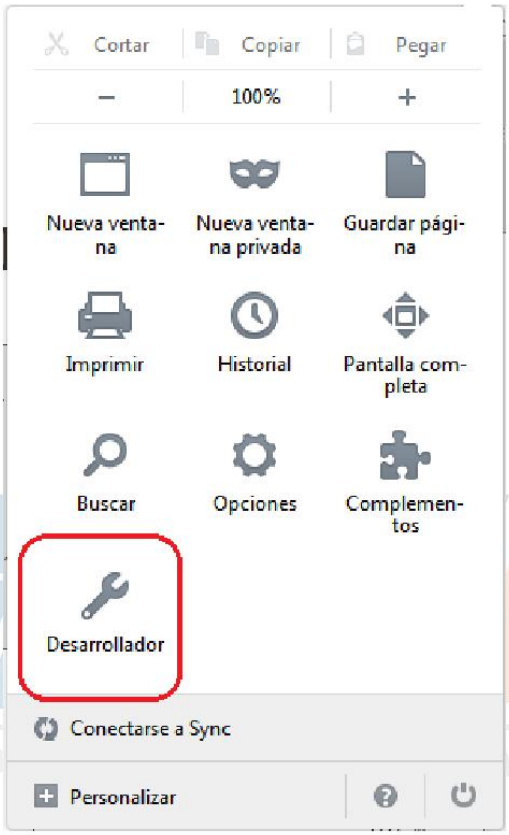
ACADEMIA DE SOFTWARE

b. Respuesta a un evento
Es la otra manera de ejecutar scripts, pero antes de verla debemos hablar sobre los eventos. Las acciones que queremos realizar como respuesta a un evento se han de indicar dentro del mismo código HTML, pero en este caso se indican en atributos HTML que se colocan dentro de la etiqueta que queremos que responda a las acciones del usuario.

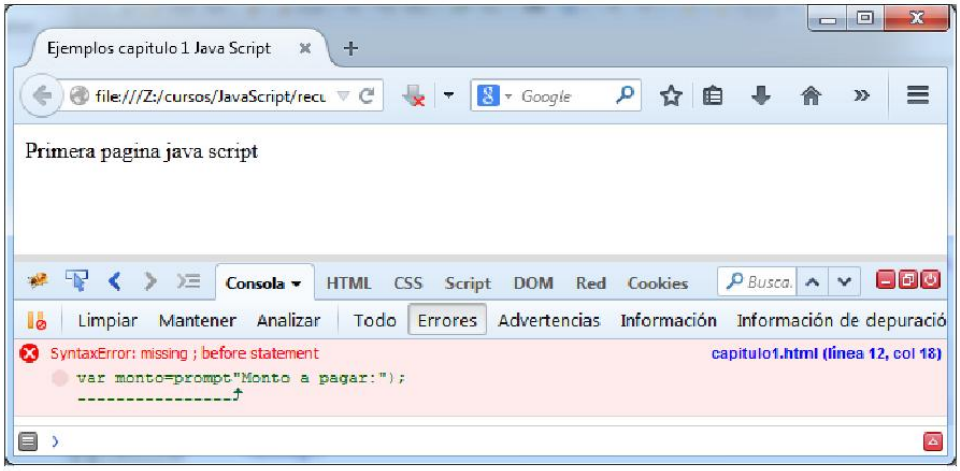
1.3.- Detección de Errores

Es muy común en el desarrollo de aplicaciones cometer errores, ya sea en la transcripción (errores de sintaxis) o en la lógica (errores lógicos). Como el encargado de ejecutar el código JavaScript es el navegador, es éste quien se encarga de ofrecer al programador alguna herramienta para detectar errores. Por lo tanto, la forma de detectar errores depende del navegador. Si se está usando Firefox, la versión 33 posee en el menú de

herramientas una opción denominada "Desarrollador", como se muestra en la imagen:



Firefox posee una herramienta llamada "Firebug" que permite entre otras cosas visualizar los errores detectados por el navegador. La primera señal evidente de que hay algún error en el código es que no se ejecutan los script o algunos de ellos. Firebug se puede iniciar presionando la tecla F12. En la pestaña errores muestra la línea donde está el error (siempre y cuando sea de sintaxis o que se haga referencia a algo que no existe):


Ejemplos capitulo 1 Java Script
file:///Z:/cursos/JavaScript/recu
Google
Primera pagina java script
Consola
HTML
CSS
Script
DOM
Red
Cookies
Busca
Limpiar
Mantener
Analizar
Todo
Errores
Advertencias
Información
Información de depuración
SyntaxError: missing ; before statement
capitulo1.html (línea 12, col 18)
var monto=prompt"Monto a pagar:");

Capítulo 2. PROGRAMACIÓN BÁSICA

2.1.- Ubicación Del Código Java Scripts

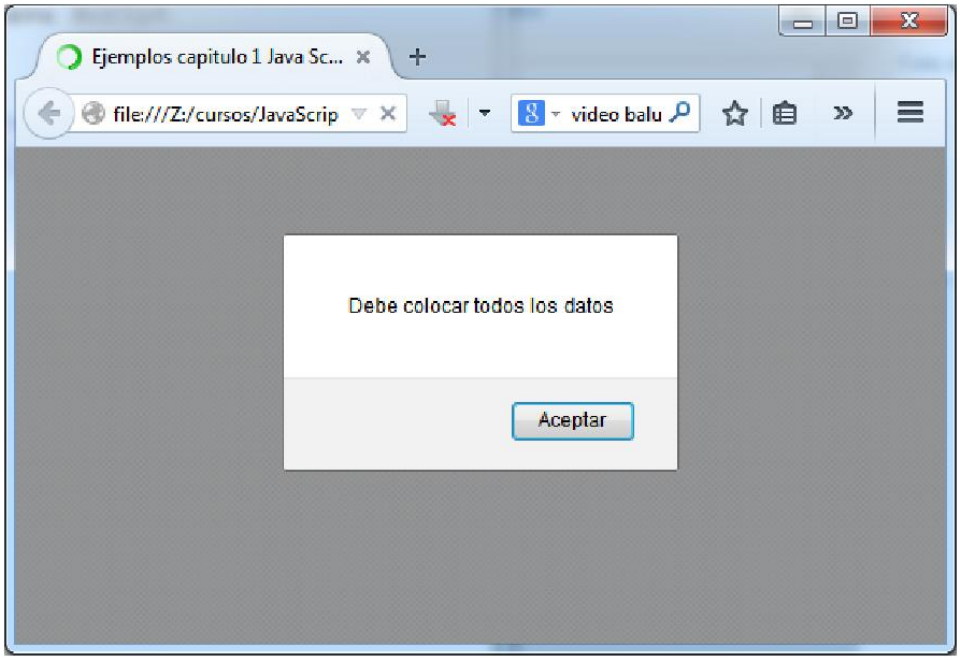
EL código Javascript se coloca dentro del propio documento HTML. Esto quiere decir que en la página se mezclan los dos lenguajes de programación, y para que estos dos lenguajes se puedan mezclar sin problemas se han de incluir unos delimitadores que separan las etiquetas HTML de las instrucciones Javascript. La siguiente imagen muestra un ejemplo:

```
    <head>
        <title>
            Ejemplos capitulo 1 Java Script
        </title>
    </head>
    <script language="javascript">
        // este es un bloque de javascript
    </script>
    <body>

    </body>
    <script language="javascript">
        /* este es otr bloque de javascript independiente
            del primero */
    </script>
</html>
```

Estos delimitadores son las etiquetas <SCRIPT> y </SCRIPT>. Todo el código Javascript que se coloque en la página ha de ser introducido entre estas dos etiquetas. Se pueden colocar la cantidad de bloques de script que hagan falta y donde sea necesario. No hay un bloque del HTML donde sea obligatorio colocar los bloques de script, pero típicamente se colocan en el head o antes del body. La ubicación del script puede ser determinante para la apropiada ejecución del mismo.

2.2.- Mensajes Emergentes

Una de las tareas que más comúnmente se realizan con Java Script es notificarle al usuario final la ocurrencia de algún evento con una ventana emergente. En la siguiente imagen se muestra un ejemplo de como aparecen las ventanas emergentes:



Estas ventanas emergentes hay que tratarlas con cuidado, debido a que en algún momento se utilizaron de forma negativa, por lo que las nuevas versiones de los navegadores bloquearon por defecto estas ventanas. La instrucción para mostrar mensajes emergentes se llama "alert". La siguiente imagen muestra un ejemplo de como se usa la función alert:

```
<html>
    <head>
        <title>
            Ejemplos capitulo 1 Java Script
        </title>
    </head>
    <script language="javascript">
        // codigo de ejecucion directa
        alert("Debe colocar todos los datos");
    </script>
    <body>
        Primera pagina java script
    </body>
</html>
```

2.3.- Archivos Externos de Javascript

Otra manera de incluir scripts en páginas web, implementada a partir de Javascript 1.1, es incluir archivos externos donde se pueden colocar muchas funciones que se utilicen en la página. Esta práctica se realiza generalmente cuando hay mucho código JavaScript y no se quiere abarrotar el código HTML con éste, o cuando hay código que se desea reutilizar en varias páginas. Los archivos deben tener extensión .js y en ellos no hace falta colocar las etiquetas <script>. Es una buena práctica que todos los archivos js estén en una misma carpeta. La siguiente imagen muestra un ejemplo de un archivo js:

```
capitulo1.html   validaciones.js
1  // validaciones
2
3  alert("debe colocar todos los datos");
```

Para hacer uso del código que se encuentra en el archivo externo, se debe incluir haciendo uso de la misma etiqueta de script pero usando la propiedad "src". Generalmente se incluyen en el head de la página. Se

deben colocar tantas etiquetas script como archivos se desean incluir. Una particularidad es que no se puede colocar código java script dentro de un par de etiquetas que están incluyendo un archivo externo, debido a que ese código no se ejecutará. En el siguiente ejemplo se muestra como incluir 2 archivos externos:

```
<html>
    <head>
        <title>
            Ejemplos capitulo 1 Java Script
        </title>
        <script language="javascript" src="ejemplo1.js"/></script>
        <script language="javascript" src="validaciones.js"/></script>
    </head>
    <script language="javascript">
        // codigo de ejecucion directa
        alert("hola mundo");
    </script>
    <body>
        Primera pagina java script
    </body>
```

El orden en que se incluyan los archivos externos de scripts puede ser irrelevante o no, dependerá de si un archivo utiliza lo que está declarado en el otro. Por ejemplo, existe una librería de Bootstrap que se basa en JQuery, en este caso, primero debe incluirse el archivo de jquery.js y luego el archivo bootstrap.js para que se pueda ejecutar correctamente la librería de Bootstrap.

Capítulo 3. VARIABLES Y TIPOS DE DATOS

3.1.- Manejo de Variables

En JavaScript se manejan variables de diversas clases de información, como textos o números. Cada una de estas clases de información son los tipos de datos. Javascript distingue entre los tipos de datos:

- Numérico
- Booleano
- Cadenas
- Objetos
- Arreglos

Los nombres de las variables han de construirse con caracteres alfanuméricos y/o el carácter subrayado (_). Aparte de esta, hay una serie de reglas adicionales para construir nombres para variables. La más importante es que tienen que comenzar por una letra o el subrayado. No se puede utilizar caracteres especiales u operadores aritméticos (+, $, ?). Algunos ejemplos de nombres admitidos para las variables podrían ser:

```
Edad
paisDeNacimiento
_nombre
```

ACADEMIA DE SOFTWARE

Tampoco se pueden utilizar palabras reservadas del lenguaje.

Es una costumbre habitual en los lenguajes de programación definir las variables que se van a usar en los programas. En JavaScript no es obligatorio declarar las variables, aunque es recomendable hacerlo. La forma de declarar variables es:

```
var operando1 ;
var operando2 ;
```

También se puede asignar un valor a la variable cuando se está declarando:

```
var operando1 = 23 ;
```

var operando2 = 33 ;

Se puede mostrar el valor de una variable usando "alert". Cuando una variable no ha tomado valor tiene por defecto el valor "undefined".

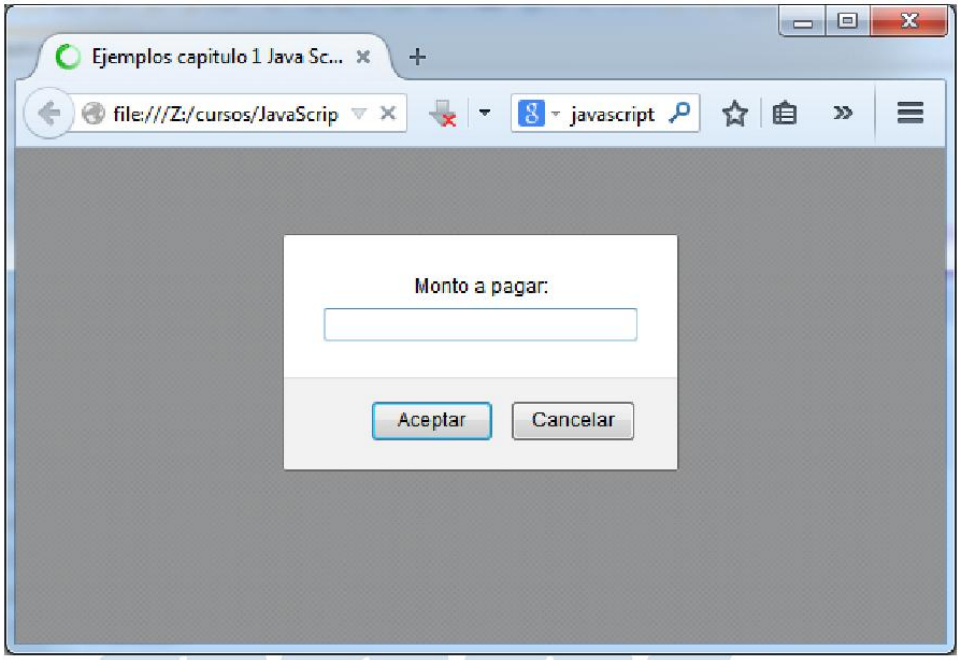
3.2.- Operadores

Los operadores de JavaScript son similares a los de lenguaje C. Como en todos los lenguajes de programación, JavaScript posee operadores de asignación, aritméticos, lógicos y relacionales. La siguiente tabla muestra un resumen de los operadores aritméticos:

Operador	Significado	Ejemplo
+	suma	números y cadenas
-	resta	
*	producto	
/	división	
%	módulo (resto)	20 % 10 (= 0)
++	suma tipográfica	variable++; ++variable; (variable = variable + 1)
-- (dos guiones)	resta tipográfica	variable--; --variable; (variable = variable - 1)

3.3.- La Ventana Prompt

En ocasiones se necesita solicitar algún dato al usuario. Para esto JavaScript posee una función llamada "prompt", que muestra una ventana con un mensaje y un cuadro de texto para que el usuario introduzca información. La siguiente imagen muestra como aparece la ventana solicitando información al usuario:

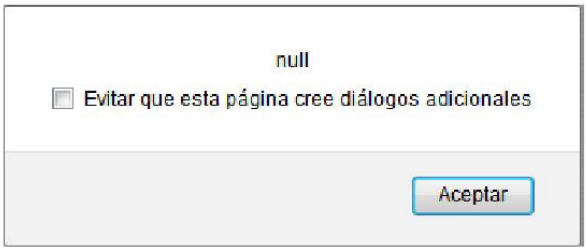


El dato que el usuario escribe puede almacenarse en una variable para luego ser utilizada. Es importante resaltar que el dato que retorna "prompt" es de tipo String, por lo tanto, debe tener en cuenta que si el dato solicitado se quiere manipular como número, debe convertirse. En la imagen se muestra un ejemplo de cómo usar la función "prompt":

```
        <script language="javascript" src="ejemplo1.js"/></script>
        <script language="javascript" src="validaciones.js"/></script>
    </head>
    <script language="javascript">
        // codigo de ejecucion directa
        //alert("hola mundo");
        var monto=prompt("Monto a pagar:");
        var nro_horas=prompt("Numero de horas:");
    </script>
    <body>
        Primera pagina java script
    </body>
</html>
```

Si el usuario hace click en el botón "Aceptar" sin escribir nada, prompt retornará una cadena vacía. Si el usuario hace click en el botón "Cancelar", la función prompt retornará el valor "null":

```
<script language="javascript">
    // codigo de ejecucion directa
    var monto=prompt("Monto a pagar:");

    alert(monto);
</script>
```



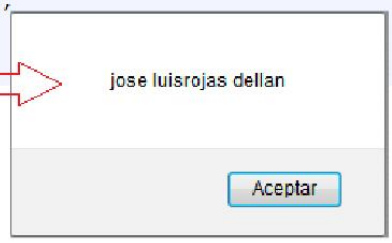
CADI F1
ACADEMIA DE SOFTWARE

Capítulo 4. MANIPULACIÓN DE STRINGS

4.1.- La Clase Strings

JavaScript posee una clase implícita llamada String que es utilizada para manipular las cadenas de caracteres. Una cadena de caracteres va encerrada entre comillas dobles o simples. En la siguiente imagen se muestra un ejemplo de cómo declarar variables que contienen valores String. Uno de las operaciones más comunes con los strings es concatenarlos (unir 2 cadenas o más). Esto se logra con el operador "+":

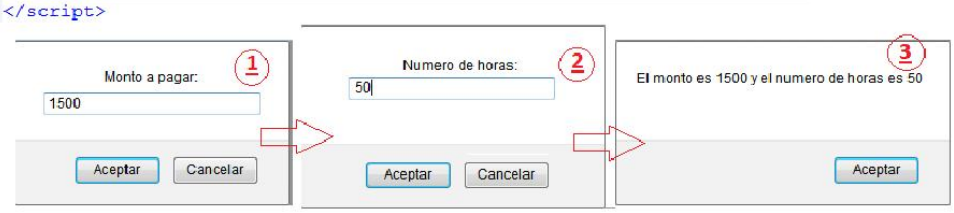
```
    <script language="javascript">
        // codigo de ejecucion directa
        var nombre="jose luis";
        var apellido='rojas dellan';

        alert(nombre + apellido);
    </script>
    <body>
        Primera pagina java script
    </body>
</html>
```



ACADEMIA DE SOFTWARE

Se pueden concatenar varias cadenas a la vez:

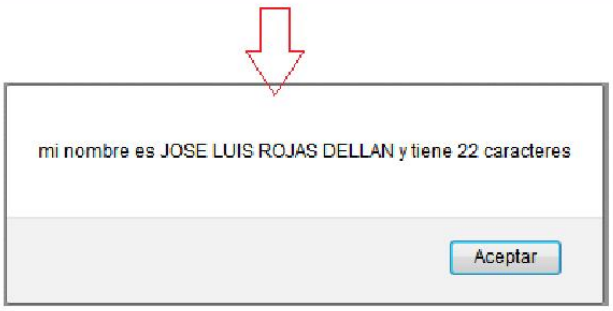
```
<script language="javascript">
    // codigo de ejecucion directa
    //alert("hola mundo");
    var monto=prompt("Monto a pagar:");
    var nro_horas=prompt("Numero de horas:");

    alert("El monto es "+monto+" y el numero de horas es "+nro_horas);
</script>
```



4.2.- Métodos de la Clase String

Otra de las operaciones muy comunes con los string es convertirlos a mayúscula. Para esto se utiliza el método toUpperCase() de la clase string. El método toLowerCase() convierte a minúsculas. También es muy común necesitar conocer la cantidad de caracteres que tiene una cadena. Esto se logra con la propiedad "length". En la siguiente imagen se muestra un ejemplo de cómo usar estas funciones:

```
<script language="javascript">
    // codigo de ejecucion directa
    var nombre="jose luis";
    var apellido='rojas dellan';
    var nombrecompleto=nombre + " " + apellido;
    alert("mi nombre es "+nombrecompleto.toUpperCase()+
          " y tiene "+nombrecompleto.length+" caracteres");
</script>
```



4.3.- Conversión de String

También es muy común necesitar convertir datos string a números, ya sean enteros o reales. Para ellos JavaScript tiene 2 funciones: parseInt, convierte de string a un número entero y parseFloat, que convierte de string a un número real. El siguiente ejemplo muestra el uso de ambas funciones:

```
<script language="javascript">
    // codigo de ejecucion directa
    var dato1=prompt("Monto a pagar:");
    var dato2=prompt("Numero de horas:");

    monto=parseFloat(dato1);
    nro_horas=parseInt(dato2);
    bono=monto*0.1;
    horas_extras=nro_horas-10;

    alert("El bono es "+bono+
          " y el numero de horas extras "+nro_horas);
</script>
```

Si la cadena que se está intentando convertir a número no es un número, las funciones de conversión retornan el valor NaN (Not a Number). Esto indica que la conversión no pudo realizarse exitosamente. Al realizar cualquier operación aritmética con un valor NaN el resultado será un NaN.

ACADEMIA DE SOFTWARE

Capítulo 5. INSTRUCCIONES CONDICIONALES

5.1.- Condicionales

Antes de hacer una condición deben conocerse los operadores relacionales, que permiten comparar una variable con otra o con un valor literal. Los operadores relacionales se muestran en la siguiente tabla:

Operador	Significado	Ejemplo
==	es igual a	5 == 8 es falso
!=	no es igual a	5 != 1 es verdad
>	es mayor que	5 > 1 es verdad
<	es menor que	5 < 8 es verdad
>=	es mayor o igual que	5 >= 8 es falso
<=	es menor o igual que	5 <= 1 es falso

IF es una estructura de control utilizada para tomar decisiones. Es un condicional que realiza unas u otras operaciones en función de una expresión. Funciona de la siguiente manera, primero se evalúa una expresión, si el resultado es verdadero, se realizan algunas instrucciones. La sintaxis de la estructura IF es la siguiente:

ACADEMIA DE SOFTWARE

```
if (expresión) {
  acciones a realizar en caso positivo
}
```

La siguiente imagen muestra un ejemplo de cómo usar el if:

```
<script language="javascript">
    // codigo de ejecucion directa
    var monto=prompt("Monto a pagar:");

    if (monto==null)
    {
        alert("No escribio el monto");
        monto=0;
    }
    alert("El monto es "+monto);
</script>
```

Opcionalmente se pueden indicar acciones a realizar en caso de que la evaluación de la sentencia sea falsa:

```
if (expresión) {
  acciones a realizar en caso positivo
}else {
  acciones a realizar en caso negativo
}
```

```
<script language="javascript">
    // codigo de ejecucion directa
    var monto=prompt("Monto a pagar:");

    if (monto==null)
        alert("El usuario hizo click en cancelar");
    else
        if (monto.length==0)
            alert("El usuario no introdujo nada");
        else
            alert("El monto es "+monto);
</script>
```

Los operadores lógicos ("and" y "or") permiten crear expresiones lógicas más complejas. En la siguiente tabla se listan los operadores lógicos

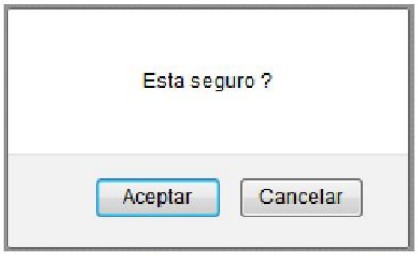
Operador	Significado	Ejemplo
&&	Y	1 == 1 && 2 < 1 es falso
||	O	1 == 2 || 15 > 2 es verdad
!	NO	!(1 > 2) es verdad

5.2.- Confirm

JavaScript posee una función que permite preguntar al usuario si desea realizar una acción o no. Muestra una ventana como "alert" pero con los botones "Aceptar" y "Cancelar". Si el usuario hace click en el botón "Aceptar" retorna true, en caso contrario, retorna false. Cómo implica una

decisión, generalmente va acompañado de un condicional. En la siguiente imagen se muestra un ejemplo de cómo usar el confirm:

```
<script language="javascript">
    // codigo de ejecucion directa

    if (confirm("Esta seguro ?"))
        alert("El usuario hizo click en aceptar");
    else
        alert("El usuario hizo click en cancelar");
</script>
```



ACADEMIA DE SOFTWARE

5.3.- Funciones Especiales

JavaScript tiene algunas funciones de utilidad para evaluar el estado de algunas variables. En ocasiones se necesita verificar si una variable contiene un valor numérico válido (NaN). Para ello se utiliza la función "isNaN". Esta función devuelve "true" si la variable no es un número válido y devuelve "false" si la variable contiene un número válido. Un ejemplo de como usar la función isNaN es el siguiente:

```
<script language="javascript">
    // codigo de ejecucion directa
    var edad= 10;

    if (isNaN(edad))
        alert("No es un número");
    else
        alert("Es un número válido ");
</script>
```

Esta función es usada especialmente cuando los valores numéricos son entradas introducidas por el usuario, debido a que el usuario puede cometer un error y colocar un valor inválido. Si el valor de la variable es leído con la función "prompt", se puede evaluar si al convertir a número el string leído se convierte exitosamente, en cuyo caso, se obtiene el valor numérico, en caso contrario, se obtiene un NaN. El siguiente es un ejemplo de como combinar IsNaN con prompt:

```
<script language="javascript">
    // codigo de ejecucion directa
    var monto=prompt("Monto a pagar:");

    if (isNaN(monto))
        alert("El usuario introdujo un valor inválido");
    else
        alert("El monto es "+monto);
</script>
```

Capítulo 6. CICLOS

6.1.- For

Java Script como cualquier otro lenguaje de programación posee instrucciones para realizar ciclos repetitivos. Uno de ellos es el ciclo for. El ciclo for se utiliza para repetir más instrucciones un determinado número de veces. De entre todos los bucles, el FOR se suele utilizar cuando sabemos con seguridad el número de veces que queremos que se ejecute la sentencia. La sintaxis del bucle se muestra a continuación:

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

En la imagen se muestra un ejemplo de cómo usar el ciclo for:

```
<script language="javascript">
    // codigo de ejecucion directa
    var nro_alumnos=prompt("Introduzca el numero de alumnos:");
    var total_nota=0;

    for (i=1;i<=nro_alumnos;i++)
    {
        var nota=parseFloat(prompt("Introduzca la nota del alumno "+i));
        total_nota+=nota;
    }
    promedio=total_nota/nro_alumnos;
    alert("El promedio de la seccion:"+promedio);

</script>
```

6.2.- While

Otro de los tipos de ciclos es el WHILE. Este tipo de ciclo se utilizan cuando se quiere 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 la condición que se tiene que cumplir para que se realice una iteración. La sintaxis general del while es la siguiente:

```
while ( expresión lógica )
{
   sentencias a ejecutar
}
```

En la siguiente imagen se muestra un ejemplo del uso del while:

```
<script language="javascript">
    var total_nota=0;
    var nro=0;
    while (confirm("Desea registrar una nota?"))
    {
        var nota=parseInt(prompt("Introduzca la nota del alumno "+(nro+1)));
        if (isNaN(nota))
            alert("No introdujo una nota válida");
        else
        {
            total_nota+=parseFloat(nota);
            nro_alumnos++;
        }
    }
    if (nro_alumnos>0)
    {
        promedio=total_nota/nro_alumnos;
        alert("Se registraron "+nro_alumnos+". El promedio es:"+promedio);
    }else
        alert("No se registraron notas");
</script>
```

ACADEMIA DE SOFTWARE

6.3.- Do While

Otro de los ciclos que posee JavaScript es el do..while, que se utiliza cuando se necesita que el ciclo se ejecute al menos una vez. La diferencia con el ciclo "while" es que la condición se evalúa al final. Si la expresión evaluada es verdadera, el ciclo se repite una vez más, en caso contrario, se rompe el ciclo. El uso general del ciclo "do while" es:

```
do
{
.....
}while ( expresión lógica )
```

El mismo ejemplo realizado con el ciclo "while" se puede hacer con "do..while":

```
<script language="javascript">
    var total_nota=0;
    var nro=0;
    do
    {
        var nota=parseInt(prompt("Introduzca la nota del alumno "+(nro+1)));
        if (isNaN(nota))
            alert("No introdujo una nota válida");
        else
        {
            total_nota+=parseFloat(nota);
            nro_alumnos++;
        }
    }while (confirm("Desea registrar otra nota?"));
    if (nro_alumnos>0)
    {
        promedio=total_nota/nro_alumnos;
        alert("Se registraron "+nro_alumnos+". El promedio es:"+promedio);
    }else
        alert("No se registraron notas");
</script>
```

Un ejemplo muy común para usar el ciclo "do..while" es para validar los valores leídos por el teclado con "prompt". Al leer el valor, se evalúa si contiene un valor válido, en caso contrario, se vuelve a leer el valor. En el siguiente ejemplo, se lee el valor y trata de convertir a float (usando parseFloat), si no se logra convertir, la función retornará NaN. Si el valor isNaN, se debe volver a leer:

ACADEMIA DE SOFTWARE

```
<script  type="text/javascript" >
    do
    {
        nota=parseFloat(prompt("Introduzca la nota del alumno "));
        if (isNaN(nota))
            alert("No introdujo una nota válida");
    }while (isNaN(nota));
    alert("La nota leida fue: "+nota);
</script>
```

Capítulo 7. FUNCIONES

7.1.- Definición de Funciones

En ocasiones se tienen muchas instrucciones y deben agruparse en bloques. También en ocasiones hace un bloque de código debe reutilizarse. Para ellos se utilizan las funciones. Una función se debe definir con una sintaxis especial que vamos a conocer a continuación:

```
function nombrefuncion (){
        instrucciones de la función
        ...
}
```

En principio, se pueden colocar las funciones en cualquier parte de la página, siempre entre etiquetas <SCRIPT>. No obstante existe una limitación a la hora de colocarla con relación a los lugares desde donde se la llame. Lo más normal es colocar la función antes de cualquier llamada a la misma.

El código que está adentro de una función no se ejecutará a menos que en algún lugar se llame la función. La siguiente imagen muestra un ejemplo de una función:

ACADEMIA DE SOFTWARE

```
<script language="javascript">
    function saludar()
    {
        alert("Esto es una funcion");
        if (confirm())
            alert("respondio que si");
        else
            alert("respondio que no");
    }
</script>
```

7.2.- Parámetros

Las funciones pueden tener parámetros, los cuales se colocan entre los paréntesis en la declaración de la función. JavaScript no maneja los tipos de datos explícitamente, por lo tanto, no es necesario indicar el tipo de dato de los parámetros. En la siguiente imagen se muestra un ejemplo del uso de los parámetros y de cómo debe llamarse la función pasándole parámetros a la misma:

```
<script language="javascript">
    function saludar(nro, monto)
    {
        alert("El numero es "+nro+" y el monto es "+
              monto+" bs");
    }
    saludar(5,1500);
</script>
```

Las funciones pueden retornar un valor. Esto es muy frecuente cuando se necesita que calculen algo. El valor que estas retornan puede almacenarse en una variable para luego ser utilizada. El siguiente es un ejemplo de usar el valor de retorno de una función:

ACADEMIA DE SOFTWARE

```
<script language="javascript">
    function calcular_bono(nro, monto)
    {
        if (nro>0)
            return monto*0.5;
        else
            return monto*0.1;
    }
    var bono= calcular_bono(5,1500);
    alert("El bono es de "+bono+" bs");
</script>
```

Capítulo 8. EVENTOS

8.1.- Manejadores de Eventos

En el primer capítulo se mencionó que el código JavaScript se puede ejecutar directo o como respuesta a un evento. Hasta ahora se ha visto el código de ejecución directa. Ahora se explicará el código de ejecución como respuesta a eventos.

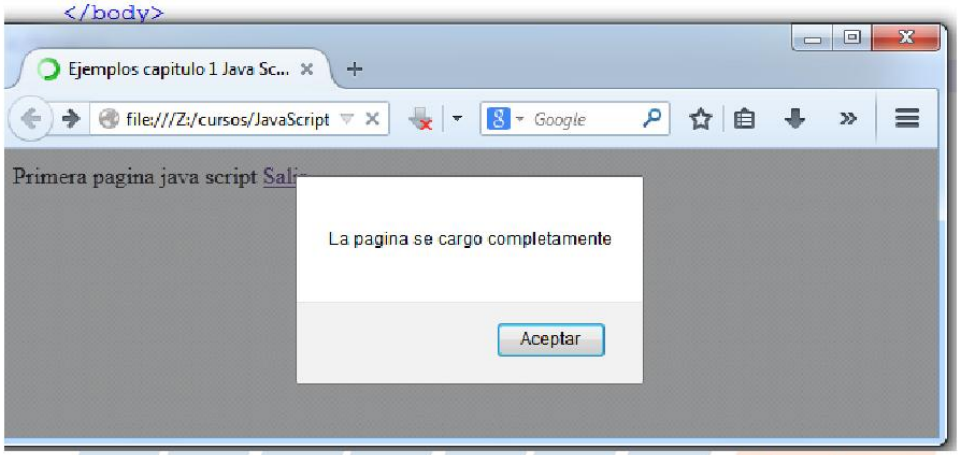
Los eventos son la manera que se tiene para controlar las acciones de los visitantes y definir un comportamiento de la página cuando se produzcan. Cuando un usuario visita una página web e interactúa con ella se producen los eventos y con Javascript se puede definir qué se necesita que ocurra cuando se produzcan.

Para definir las acciones que se quieren realizar al producirse un evento se utilizan los manejadores de eventos. Existen muchos tipos de manejadores de eventos, para muchos tipos de acciones del usuario. El manejador de eventos se coloca en la etiqueta HTML del elemento de la página que queremos que responda a las acciones del usuario.

Casi todas las etiquetas HTML pueden responder a eventos. Algunas pueden responder a algunos eventos y otros no. Se pueden programar varios eventos al mismo elemento HTML. Por ejemplo, a la etiqueta body se le puede programar el evento "onload" y "onunload". El evento "onload" se dispara cuando la página termina de cargarse en el navegador. El evento "onunload" se dispara cuando el visitante abandona la página.

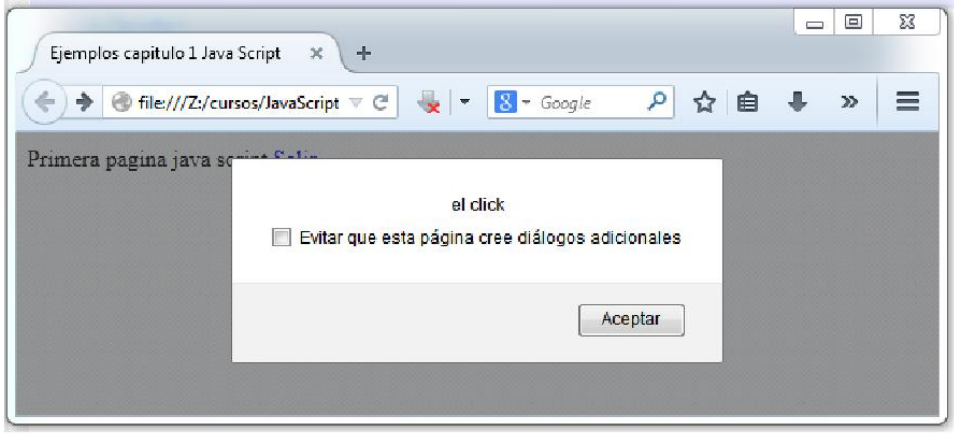
La siguiente imagen muestra un ejemplo de como programar estos eventos:

```
<script language="javascript">
    // codigo de ejecucion directa

</script>
<body onload="alert('La pagina se cargo completamente')"
      onunload="alert('adios')">
    Primera pagina java script
    <a href="http://www.cadif1.com">Salir</a>
</body>
```



Dentro de los manejadores de eventos se pueden colocar tantas instrucciones como hagan falta, separadas por punto y coma. Lo habitual es colocar una sola instrucción. Otro de los eventos que se programa muy frecuentemente es el "onclick". La etiqueta "href" (etiqueta para hipervínculos) puede responder a este evento. La siguiente imagen muestra un ejemplo de como programar el evento onclick:

```
Primera pagina java script
<a href="#" onclick="alert('el click')">Salir</a>
```



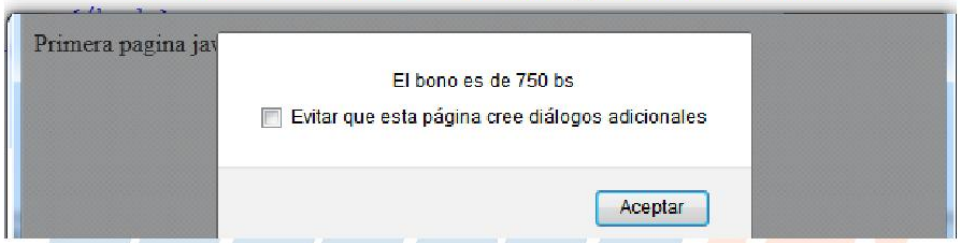
Existen muchos otros eventos. A continuación se describen algunos de ellos y las etiquetas que pueden responder a estos:

- onfocus: ocurre cuando un objeto toma el foco.
- onblur: ocurre cuando un objeto pierde el foco.
- onkeypress: ocurre cuando se presiona una tecla.
- onmousemove: ocurre cuando el cursor del ratón se mueve sobre un objeto.
- onmouseover: ocurre cuando el cursor del ratón se coloca sobre un objeto.
- onsubmit: cuando se intenta enviar un formulario.
- onscroll: cuando se hace scroll en la ventana del navegador.
- onchange: cuando el contenido de un elemento cambia.

8.2.- Eventos y Funciones

Colocar muchas instrucciones separadas con punto y coma (;) directamente en la etiqueta HTML para manejar un evento no es lo más recomendable. La mejor práctica es colocar en el HTML la llamada a una función. Por esto, combinar eventos y funciones es muy habitual. Veamos un ejemplo de como crear una ventana con una página nueva a partir puro código javascript:

```
<script language="javascript">
    function calcular_bono()
    {
        var nro=prompt("Numero:");
        var monto=prompt("Monto:");
        if (nro>0)
            bono=monto*0.5;
        else
            bono=monto*0.1;

        alert("El bono es de "+bono+" bs");
    }
</script>
<body>
    Primera pagina java script
    <a href="#" onclick="calcular_bono()">Calcular</a>
```



Cualquiera de los eventos puede llamar a una función. El siguiente es un ejemplo de cómo llamar a una función desde el evento onload y del evento keypress de un input, el cual tiene un parámetro llamado "event" que le envía la información de la tecla presionada:

```
<script language="javascript">
    function calcular_bono()
    {
        var nro=prompt("Numero:");
        var monto=prompt("Monto:");
        if (nro>0)
            bono=monto*0.5;
        else
            bono=monto*0.1;

        alert("El bono es de "+bono+" bs");
    }
    function mostrar_tecla(tecla)
    {
        alert("presion la tecla:"+tecla.key);
    }
</script>
<body onload="calcular_bono()">
    Primera pagina java script
    <input type="text" onkeypress="mostrar_tecla(event)" />
</body>
```

ACADEMIA DE SOFTWARE

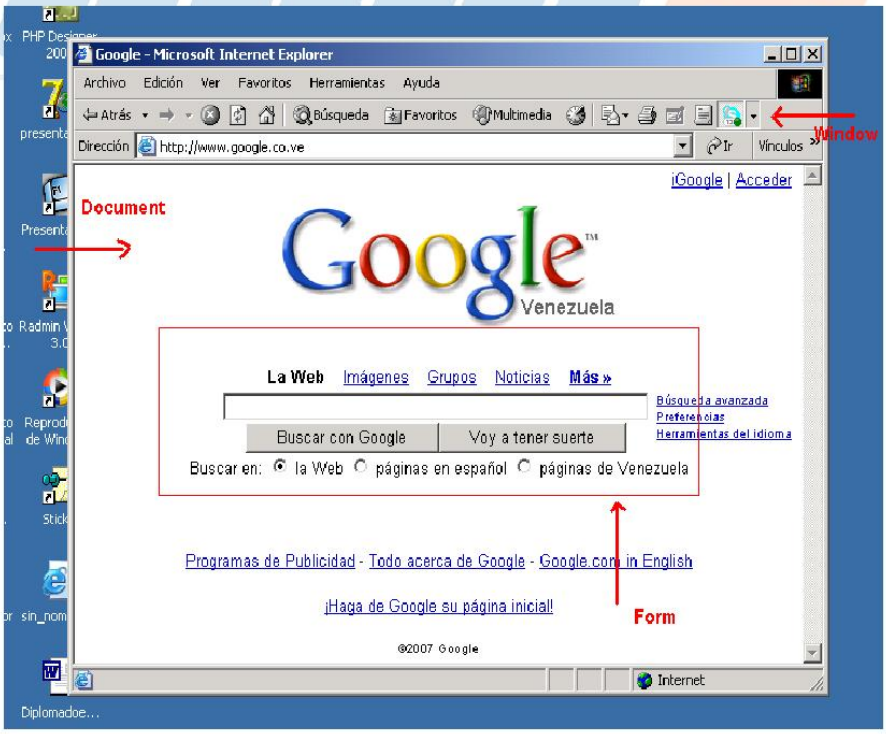
Capítulo 9. DOM - OBJETO WINDOW

9.1.- ¿Qué es la jerarquía de objetos?

Cuando se carga una página, el navegador crea una jerarquía de objetos en memoria llamado DOM (Document Object Model), que sirven para controlar los distintos elementos de dicha página. Con Javascript se puede manipular esa jerarquía de objetos, acceder a sus propiedades e invocar sus métodos.

Cualquier elemento de la página se puede controlar de una manera u otra accediendo a esa jerarquía. Es crucial conocerla bien para poder controlar perfectamente las páginas web con Javascript o cualquier otro lenguaje de programación del lado del cliente.

Los tres objetos más importantes de la jerarquía son: Window, Document y Form. La ventana del navegador (Window) contiene una pagina (Document), una página contiene formularios (Form) y un formulario contiene distintos objetos de captura de datos.



9.2.- El Objeto Window

Es el objeto principal en la jerarquía. Contiene las propiedades y métodos para controlar la ventana del navegador. De él dependen todos los demás objetos de la jerarquía. A continuación podemos ver las propiedades del objeto window:

- document: es la página que esta cargada en el momento.
- history: el historial de páginas visitadas en esa ventana.
- location: la dirección de la página que esta cargada en el momento.
- menubar: la barra de menú de navegador.
- locationbar: la barra de direcciones del navegador.
- statusbar: la barra de estado del navegador.
- frames: arreglo de marcos de la página (si tiene).
- screen: permite conocer el estado de la pantalla.
- innerHeight: la altura de la ventana del navegador.
- innerWidth: el ancho de la ventana del navegador.
- navigator: permite conocer datos del navegador.

También tiene unos métodos (o procedimientos) que permiten ejecutar acciones sobre la ventana del navegador. Los más importantes son:

- alert: muestra un mensaje estilo de aviso.
- confirm: muestra un dialogo con los botones Aceptar y Cancelar.
- close: cierra la ventana del navegador.
- home: carga la URL definida como HOME en el navegador.
- print: Imprime el contenido de la ventana.
- open: abre una nueva ventana del navegador.
- moveTo: mueve la ventana a las coordenadas especificadas (solo funciona con ventanas abiertas con open).
- resizeTo: redimensiona la ventana a los nuevos valores de ancho y alto (sólo funciona con ventanas abiertas con open).

Ejemplos de como usar estos métodos y propiedades del objeto Window son:

```
<script language="javascript">

window.resizeTo (400, 400); // cambia la ventana del navegador a 400x400
window.print();  // hace que aparezca el dialogo para imprimir
window.location = "http://www.google.com";  // hace que se cargue google
window.open("http://www.cantv.net","","" ); // abre otra ventana del navegador
window.close(); // cierra la ventana del navegardor

</script>
```

9.3.- El Método Open

Uno de los métodos más usados es el "open", que permite abrir una ventana del navegador. La forma general de llamar al método open es:

window.open(url, name, specs, replace)

El método open tiene 3 parámetros:

- la url del documento que se necesita cargar. Puede ser un archivo del sitio o un archivo de otro sitio.
- name: es un parámetro opcional que indica el nombre de la ventana o la forma en que se va a abrir la ventana, si es una ventana nueva se usa "_blank", si es la misma ventana se utiliza "".
- specs: es un parámetro opcional, que establece parámetros del aspecto de la ventana que se abrirá.
- replace: es un parámetro opcional.

Un ejemplo de uso de open es:

window.open("http://www.google.com", "","width=200,height=100");

Capítulo 10. DOM - DOCUMENT

10.1.- El Objeto Document

Con el objeto document se controla la página web y todos los elementos que contiene. El objeto document es la página actual que se está visualizando en ese momento. Depende del objeto window, pero también puede depender del objeto frames en caso de que la página se esté mostrando en un conjunto de marcos.

Normalmente no se manipula tanto como el objeto Window, pero al igual que el Window tiene propiedades y métodos para manipularlo. Las propiedades más usadas son:

- forms: un arreglo con los formularios que tiene el documento.
- parent: el objeto que contiene al documento. El window o un frame.
- bgColor: el color de fondo del documento.
- title: el título de la página.
- lastModified: fecha de la última modificación del documento.
- fgColor: Color por defecto del texto.
- linkColor: Color de los enlaces.
- domain: Nombre del dominio del servidor de la página.

Aunque poco común, con los métodos del objeto form podemos escribir información dinámicamente en la página. Para esto vamos a usar los siguientes métodos:

- open: Abre una corriente de datos para escribir en el documento (con los métodos write o writeln) .
- write: Escribe texto a la corriente de datos del documento.
- writeln: Igual que write pero finaliza el texto con un retorno de carro.
- close: Cierra una corriente de datos abierta por el método open y hace que se muestren todos los elementos.

10.2.- Buscar Elementos

Todos los elementos (etiquetas) que están en el body de un documento html, son parte del "document". Es una tarea muy común manipular alguno de estos elementos en tiempo de ejecución. Para manipular un elemento del documento primero se debe tener la referencia a éste. Se puede buscar un elemento por el "id", por el "name", por el "classname" o por el "tagname". Para cada uno de los casos, el objeto "document" posee un método para buscar:

- getElementById: busca un elemento por su "id".
- getElementByName: busca un elemento por su "name".
- getElementByTagName: retorna todos los elementos que poseen un tipo de etiqueta. Por ejemplo, todos los <p>.
- getElementByClassName: retorna todos los elementos que poseen un nombre de clase CSS.

Al buscar, cada uno de los métodos retornará una referencia al elemento buscado, en caso de no encontrar el elemento, retornará valor "undefined". El siguiente es un ejemplo para buscar un elemento por su "id":

```
if (document.getElementById("idElemento") == undefined)
        alert("El elemento no existe");
else
{
        // se hace algo
}
```

10.3.- Manipular Elementos

Luego de encontrado el elemento, conociendo que tipo de elemento es, se puede manipular, ya sea sus propiedades particulares o los estilos CSS asociados a este elemento. Por ejemplo, si el elemento buscado es un " ", un "<p>" o un "<h>", se puede modificar el contenido del mismo con la propiedad "innerHTML". Por ejemplo:

```
var elemento = document.getElementById("idElemento");
if (elemento == undefined) alert("El elemento no existe");
else
```

```
{
        elemento.innerHTML = "Nuevo contenido"
}
```

Asimismo se puede modificar las propiedades del CSS de cualquier elemento, o de varios elementos (en caso de buscar por class). Para lograrlo, se utiliza el atributo "style" y luego la propiedad específica del estilo CSS que se necesita manipular. La forma general es la siguiente:

```
document.getElementById("idElemento").style.propiedad = "valor";
```

Por ejemplo, si se necesita establecer a rojo el color de fondo de un elemento "<p>" con el id "resultado", se debe utilizar la siguiente instrucción:

```
document.getElementById("resultado").style.backgroundColor = "red"
```

Otra forma de hacerlo:

```
var elemento = document.getElementById("resultado");
elemento.style.bakgroundColor = "red"
```

ACADEMIA DE SOFTWARE

Capítulo 11. DOM - FORM

11.1.- El Objeto Form

El objeto form depende en la jerarquía de objetos del objeto document. En un objeto form podemos encontrar algunos métodos y propiedades, pero lo más destacado que podremos encontrar son cada uno de los elementos del formulario. Es decir, de un formulario dependen todos los elementos que hay dentro, como pueden ser campos de texto, cajas de selección, áreas de texto, botones de radio, etc.

Para acceder a un formulario desde el objeto document podemos hacerlo de dos formas:

1. A partir de su nombre, asignado con el atributo NAME de HTML.
2. Mediante la matriz de formularios del objeto document, con el índice del formulario al que queremos acceder (Esta forma de acceder se verá en capítulos posteriores).

Por ejemplo, si tenemos un formulario como el siguiente:

```
<FORM name="f1">
<INPUT type=text name=campo1>
<INPUT type=text name=campo2>
</FORM>
```

Se puede acceder a las propiedades y elementos del formulario de las siguientes maneras: f1, document.f1 o con su índice document.forms[0] (si suponemos que es el primero de la página).

De similar manera accedemos a los elementos de un formulario, que dependen del objeto form. Podríamos acceder al campo 1 del anterior formulario de dos maneras. Con su nombre se haría así: document.f1.campo1.

El objeto form tiene propiedades para ajustar sus atributos mediante en tiempo de ejecución. Los más usados son:

- action: dirección URL a la que le mandaremos los datos del formulario.
- elements: La matriz contiene cada uno de los campos del formulario.
- method: El método por el que mandamos la información.
- target: La ventana o frame en la que está dirigido el formulario. Cuando se envía, se actualizará la ventana o frame indicado.

Estos son los métodos que podemos invocar con un formulario:

- submit: para hacer que el formulario se envia, aunque no se haya pulsado el botón de submit.
- reset: para reinicializar todos los campos del formulario, como si se hubiese pulsado el botón de reset.

11.2.- El Objeto Input

Todos los objetos pueden manipularse de la misma manera que los explicados anteriormente (Window,document y form) , poseen también propiedades y métodos para manipularlos en tiempo de ejecución a través de código Javascript.

Uno de los que más se usa es el Textbox o cuadro de texto, que es el campo que resulta de escribir la etiqueta <INPUT type="text">. Vamos a describir sus propiedades y métodos más usados:

- defaultValue: Es el valor por defecto que tiene un campo.
- form : Hace referencia al formulario.
- value: El texto que hay escrito en el campo.
- disabled: si es true indica que el cuadro de texto estará inactivo.
- focus(): es prácticamente el único método que se usa. Sirve para dar el foco al cuadro de texto.

Capítulo 12. VALIDACIÓN DE FORMULARIOS. PARTE 1

12.1.- Validación de Formularios

Uno de los mayores usos que se le da a Javascript es el de realizar validaciones del lado del cliente para así dar al usuario una respuesta más rápida y descongestionar al servidor de tratar formularios que probablemente tengan errores.

Las validaciones con Javascript no eliminan a las validaciones del lado del servidor (hechas con cualquier lenguaje del lado del servidor), debido a que las validaciones del lado del cliente pueden ser alteradas u omitidas por usuarios expertos. La idea básicamente es crear funciones Javascript en nuestra página que validen los datos introducidos por el usuario. Si los datos son válidos, se envían a la página establecida en el action, si no, se muestra un mensaje de error, con alert o en el documento y no se envían los datos.

Las validaciones de formularios se pueden hacer de 2 formas. La primera es colocándo un botón tipo "button" en el formulario y programar el evento "onclick" del botón, en el cual se llama la función de validación. Esta función tendrá las validaciones (condicionales) apropiadas. Si se cumplen las condiciones, se llama el método "submit()" del formulario, sino, se muestran errores y no se envía el formulario. A la función se le envía por parámetro el formulario al que pertenece el botón que la llama (this.form), que contiene los elementos a ser verificados. La siguiente imagen muestra un ejemplo:

```
<script language="javascript">
    function validar(form)
    {
        if ( no_se_cumplen_validaciones )
            alert("Error ....");
        else
            form.submit();
    }
</script>
<body >
    Ejemplo de validaciones
    <form name="f1" action="">
        Cedula <input type="text" name="cedula" />
        Nombre <input type="text" name="nombre" />
        <input type="button" name="btnenviar" value="Enviar" onclick="validar(this.form)"/>
    </form>
</body>
```

La otra forma es programando el evento "onsubmit" del formulario que se desea validar. En este caso, el botón debe ser tipo "submit" para que dispare el evento del formulario. En el evento, se hace el llamado a la función de validación, la cual retornará "true" si las validaciones fueron correctas o "false" si no se cumplen las condiciones. El llamado es ligeramente diferente, porque debe colocarse la palabra "return" en el llamado de la función en el evento "onsubmit", de lo contrario se enviará el formulario independientemente de lo que retorne la función:

```
<script language="javascript">
    function validar(form)
    {
        if ( condiciones_de_validacion )
        {
            alert("Error ....");
            return false;
        }else
            return true;
    }
</script>
<body >
    Ejemplo de validaciones
    <form name="f1" action="" onsubmit="return validar(this)">
        Cedula <input type="text" name="cedula" />
        Nombre <input type="text" name="nombre" />
        <input type="submit" name="btnenviar" value="Enviar"/>
    </form>
</body>
```

12.2.- Validar Cuadros de Texto

Una de las validaciones más comunes es evitar que un cuadro de texto esté vacío al momento de enviar el formulario (campos obligatorios). Para ello se debe evaluar el contenido del input, determinando si es igual a la cadena vacía o si la longitud de la cadena es 0 (la propiedad length de los string). Conociendo el DOM se pueden realizar algunas instrucciones extras, como usar "setfocus" para ubicar el cursor en el cuadro de texto que no cumple con la validación. En la siguiente imagen se muestra un ejemplo:

```
<script language="javascript">
    function validar(form)
    {
        if ( form.cedula.value == "" )
        {
            alert("La cedula es obligatorio");
            form.cedula.focus();
            return false;
        }else
            return true;
    }
</script>
<body >
    Ejemplo de validaciones
    <form name="f1" action="" onsubmit="return validar(this)">
        Cedula <input type="text" name="cedula" />
        Nombre <input type="text" name="nombre" />
        <input type="submit" name="btnenviar" value="Enviar"/>
    </form>
</body>
```

Si hace falta validar varios cuadros de texto obligatorios, se puede hacer una condición anidada:

ACADEMIA DE SOFTWARE

```
<script language="javascript">
    function validar(form)
    {
        if ( form.cedula.value == "" )
        {
            alert("La cedula es obligatorio");
            form.cedula.focus();
            return false;
        }else
            if ( form.nombre.value == "" )
            {
                alert("El nombre es obligatorio");
                form.nombre.focus();
                return false;
            }else
                return true;
    }
</script>
<body >
    Ejemplo de validaciones
    <form name="f1" action="" onsubmit="return validar(this)">
        Cedula <input type="text" name="cedula" />
        Nombre <input type="text" name="nombre" />
        <input type="submit" name="btnenviar" value="Enviar"/>
    </form>
</body>
```

ACADEMIA DE SOFTWARE

También se puede hacer una función reutilizable en caso de que se tengan muchos campos obligatorios que validar, de forma tal que se tenga que repetir menos código:

```
<script language="javascript">
    function esta_vacio(campo,mensaje)
    {
        if ( campo.value == "" )
        {
            alert(mensaje);
            campo.focus();
            return true;
        }else
            return false;
    }
    function validar(form)
    {
        if ( esta_vacio(form.cedula,"La cedula es obligatorio"))
            return false;
        else
            if (esta_vacio(form.nombre,"El nombre es obligatorio"))
                return false
            else
                return true;
    }
</script>
```

12.3.- Validar Select

Para validar los select se puede utilizar la propiedad "value" que retorna el valor de la propiedad "value" del elemento seleccionado, o la propiedad "selectedIndex", que indica la posición del elemento seleccionado. Si un select tiene la propiedad "multiple" (que indica que se pueden seleccionar varios elementos o ninguno), y el usuario no ha seleccionado algún elemento, la propiedad "selectedIndex" tomará el valor -1. La siguiente función muestra cómo evaluar un select, si hay algún elemento seleccionado, retorna true sino retorna false:

```
<script language="javascript">
    function validar_select(select)
    {
        if (select.selectedIndex ==-1)
        {
            alert("Debe seleccionar un elemento de la lista");
            select.focus();
            return false;
        }else
            return true;
    }
</script>
```

La función que valida todo el conjunto debe llamar a la función que valida el select:

```
<script language="javascript">
    function validar(form)
    {
        if ( esta_vacio(form.cedula,"La cedula es obligatorio"))
            return false;
        else
            if (validar_select(form.ciudad)==false)
                return false;
            else
                return true;
    }
</script>
<body >
    Ejemplo de validaciones
    <form name="f1" action="" onsubmit="return validar(this)">
        Cedula <input type="text" name="cedula" onkeypress="return validar_numero(event)"/>
        Nombre <input type="text" name="nombre" onkeypress="return validarletra(event)"/>
        Correo <input type="text" name="correo" />
        Ciudad <select name="ciudad" multiple>
                    <option value="0">Seleccione</option>
                    <option value="1">Barquisimeto</option>
                    <option value="2">Caracas</option>
               </select>
        <input type="submit" name="btnenviar" value="Enviar"/>
    </form>
</body>
```

Capítulo 13. VALIDACIÓN DE FORMULARIOS. PARTE 2

13.1.- Validar correo

Para validar el correo electrónico es necesario utilizar una clase que posee JavaScript para evaluar expresiones regulares. El funcionamiento de las expresiones regulares escapa del alcance de este curso, por lo que se mostrará la expresión que debe usarse para determinar si el corre cumpla con las condiciones apropiadas: tener un @, tener al menos 1 letra antes y después del @, tener un punto (.) para el dominio después del @. Se muestra el ejemplo una función que retorne true si la cadena que está escrita en el cuadro texto tiene la estructura válida de una dirección de correo:

```
function validarcorreo(correo)
{
    regx = /^([0-9a-zA-Z]([-.\w]*[0-9a-zA-Z])*@([0-9a-zA-Z][-\w]*[0-9a-zA-Z]\.)+[a-zA-Z]{2,9})$/;
    if (regx.test(correo.value))
        return true;
    else
    {
        alert("Formato de correo incorrecto");
        correo.focus();
        return false;
    }
}
```

Esta función se puede colocar en una librería para ser reutilizada en todos las páginas donde hayan formularios que soliciten correo electrónico. En la misma función de validación donde se verifican todos los campos, se puede llamar esta función. En la siguiente imagen se muestra un ejemplo:

```
<script language="javascript">
    function validar(form)
    {
        if ( esta_vacio(form.cedula,"La cedula es obligatorio"))
            return false;
        else
            if (esta_vacio(form.nombre,"El nombre es obligatorio"))
                return false
            else
                if (esta_vacio(form.correo,"El correo es obligatorio"))
                    return false
                else
                    if (validarcorreo(form.correo)==false)
                        return false;
                    else
                        return true;
    }
</script>
```

13.2.- Validar teclas

En ocasiones es necesario validar las teclas que puede y no puede recibir un cuadro de texto. Por ejemplo, si se solicita el número de hijos, este cuadro de texto solo debe recibir números. Por lo tanto, es necesario determinar cuál tecla presionó el usuario, para así determinar si se permite o no se permite. Los input pueden responder a 3 eventos de teclado:

- onkeydown: ocurre cuando el usuario está presionando la tecla.
- onkeypress: ocurre cuando el usuario presiona la tecla.
- onkeyup: ocurre cuando el usuario suelta la tecla.

Los 3 eventos se disparan en ese orden (keydown, keypress y keyup) cuando el usuario presiona una tecla. El evento onkeypress no se dispara con todas las teclas (por ejemplo: ALT, CTRL, SHIFT, ESC) en todos los navegadores. Para detectar si el usuario ha presionado una tecla se debe usar el evento onkeydown porque funciona en todos los navegadores.

Para esto, debe usarse el evento "keypress" de los input. El evento "keypress" genera un dato llamado "event", que permite, entre otras cosas, conocer la tecla presionada por el usuario. En realidad es un objeto que posee las siguientes propiedades:

ACADEMIA DE SOFTWARE

- charCode: el código ASCII de la tecla presionada. No funciona en algunos navegadores.
- keyCode: el código ASCII de la tecla presionada.
- shiftKey: indica si la tecla esta shift presionada.
- ctrlKey: indica si la tecla esta ctrl presionada.
- altKey: indica si la tecla esta alt presionada.

Si se desea conocer la tecla equivalente al código ASCII de la tecla presionada, se debe utilizar un método de la clase String que hace la conversión. Este método se llama se "fromCharCode". La siguiente función es un ejemplo de cómo hacer la conversión:

```
<script language="javascript">
        function getChar(event)
        {
            if (event.keyCode != 0)
                return String.fromCharCode(event.keyCode) ;
            else
                if (event.charCode!=0)
                    return String.fromCharCode(event.charCode);
                else
                    return null;
        }
</script>
```

Otro aspecto que se debe entender es que para rechazar una tecla que el usuario ha presionado, se debe colocar "return false" en el evento onkeypress del input. El siguiente ejemplo rechazará todas las teclas que el usuario presione:

```
<body >
    Ejemplo de validaciones
    <form name="f1" action="" onsubmit="return validar(this)">
        Cedula <input type="text" name="cedula" onkeypress="return false"/>
        Nombre <input type="text" name="nombre" />
        Correo <input type="text" name="correo" />
        <input type="submit" name="btnenviar" value="Enviar"/>
    </form>
</body>
```

Entendiendo todos estos conceptos, se puede realizar una función que valide las teclas que son permitidas para ciertos inputs, dependiendo de la naturaleza del dato. El siguiente ejemplo muestra 2 funciones, una que valida que el usuario sólo escriba letras y la otra que valida que el usuario sólo escriba números:

```
function validarletra(e)
{
    // para que funcione en cualquier navegador
    tecla = (e.keyCode!=0) ? e.keyCode : e.charCode; ;
    // para permitir backspace
    if (tecla==8) return true;
    // de define el conjunto de caracteres validas
    patron =/[A-Za-z\s]/;
    // se convierte a caracter
    te = String.fromCharCode(tecla);
    // se evalua si la tecla presionada este en el conjunto
    return patron.test(te);
}

function validar_numero(e)
{
    // para que funcione en cualquier navegador
    tecla = (e.keyCode!=0) ? e.keyCode : e.charCode;
    // para permitir backspace
    if (tecla==8) return true;
    // de define el conjunto de caracteres validas
    patron = /[0-9-]/;
    // se convierte a caracter
    te = String.fromCharCode(tecla);
    // se evalua si la tecla presionada este en el conjunto
    return patron.test(te);
}
```

En los input se debe llamar en el keypress:

```
<body >
    Ejemplo de validaciones
    <form name="f1" action="" onsubmit="return validar(this)">
        Cedula <input type="text" name="cedula" onkeypress="return validar_numero(event)"/>
        Nombre <input type="text" name="nombre" onkeypress="return validarletra(event)"/>
        Correo <input type="text" name="correo" />
        <input type="submit" name="btnenviar" value="Enviar"/>
    </form>
</body>
```

13.3.- Validar checkbox

Para validar un checkbox solo es necesario conocer la propiedad "checked", que tendrá el valor "true" si el checkbox está marcado y el

valor "false" si no está marcado. El siguiente ejemplo muestra cómo hacer referencia a esa propiedad:

```
<script language="javascript">
    function validar_checkbox(check)
    {
        if (check.checked == false)
        {
            alert("Debe seleccionar la opcion");
            return false;
        }else
            return true;
    }
</script>
```

La función de validación completa queda como la siguiente:

```
<script language="javascript">
    function validar(form)
    {
        if ( esta_vacio(form.cedula,"La cedula es obligatorio"))
            return false;
        else
            if (validarcorreo(form.correo)==false)
                return false;
            else
                if (validar_select(form.ciudad)==false)
                    return false;
                else
                    if (validar_checkbox(form.condiciones))
                        return true;
                    else
                        return false;
    }
</script>
```

Capítulo 14. MANEJO DE FECHAS

14.1.- Date

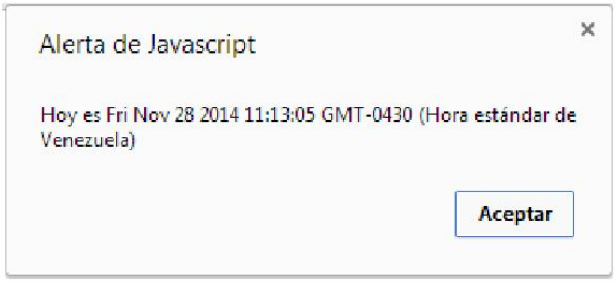
Obtener la fecha de la computadora donde se está ejecutando la página es una tarea que muy comúnmente se realiza. Para esto, se utiliza la clase Date. Esta clase al usarse posee toda la información de fecha y hora: dia, mes, año, hora, minutos, segundos y zona horaria. Para acceder a algunos de estos datos es necesario usar los métodos que posee esta clase. Los método que se pueden usar son:

- getFullYear(): obtiene el año en 4 dígitos de la fecha actual.
- getYear(): obtiene el año en 2 dígitos de la fecha actual.
- getMonth(): obtiene el mes de la fecha actual.
- getDate(): obtiene el día de la fecha actual.
- getHours(): obtiene la hora de la fecha actual.
- getMinutes(): obtiene los minutos de la fecha actual.
- getSeconds(): obtiene los segundos de la fecha actual.
- toString(): genera toda la información de la fecha y hora y lo convierte en un string.

Un ejemplo sencillo de cómo usar la clase Date es el siguiente:

ACADEMIA DE SOFTWARE

```
<script language="javascript">
    var today = new Date();

    alert("Hoy es "+today.toString());
</script>
```



Otro ejemplo de cómo usar métodos de la clase Date:

```
<script language="javascript">
    var today = new Date();

    var today = new Date();
    var h = today.getHours();
    var m = today.getMinutes();
    var s = today.getSeconds();

    alert("Son las "+h+ ":" + m + ":" + s);
</script>
```

14.2.- Settimeout

Es muy común necesitar ejecutar código automáticamente al cabo de cierto tiempo, por ejemplo: cerrar una página al pasar cierto tiempo de inactividad (como hacen las páginas de los bancos), mostrar la hora en algún lugar de la página, entre otras cosas. Para realizar esta tarea se pueden utilizar dos métodos: setTimeout y setInterval. A ambos se le indica la función que se va a ejecutar y la cantidad de milisegundos que deben transcurrir para que se ejecute la función. La diferencia entre estos métodos es que seTimeout se ejecuta una sola vez al transcurrir el tiempo indicado y setInterval se ejecuta cada vez que transcurre el tiempo indicado. La forma general de los métodos es:

window.setTimeout(funcion, milisegundos);
window.setInterval(funcion, milisegundos);

En el siguiente ejemplo se muestra el uso del método setTimeout, en el cual luego de transcurrir 5 segundos de cargarse la página se mostrará un mensaje:

```
<script language="javascript">

    function saludar()
    {
        alert("Transcurrieron 5 segundos");
    }
</script>
<body onload="window.setTimeout('saludar()',5000);">

</body>
```

Es posible detener la ejecución del setTimeout almacenando en una variable el identificador asociado al timer que se programa. En el siguiente ejemplo se muestra cómo desactivar el setTimeout al hacer click en un botón:

```
<script language="javascript">
    var timer;
    function activar()
    {
        timer=window.setTimeout('saludar()',5000);
        alert("Se activo el reloj");
    }
    function desactivar()
    {
        clearTimeout(timer);
    }
    function saludar()
    {
        alert("Transcurrieron 5 segundos");
    }
</script>
<body >
    <input type="button" value="Activar" onclick="activar()" />
    <input type="button" value="Desactivar" onclick="desactivar()" />
</body>
```

14.3.- Setinterval

El funcionamiento del setInterval es idéntico que el setTimeout. En la siguiente imagen se muestra un ejemplo de cómo usarlo:

```
<script language="javascript">
    var timer;
    var contador=0;
    function activar()
    {
        timer=window.setInterval('saludar()',5000);
        alert("Se activo el reloj");
    }
    function desactivar()
    {
        clearInterval(timer);
    }
    function saludar()
    {
        contador++;
        alert("Se ha ejecutado "+contador+" veces");
    }
</script>
<body >
    <input type="button" value="Activar" onclick="activar()" />
    <input type="button" value="Desactivar" onclick="desactivar()" />
</body>
```

ACADEMIA DE SOFTWARE