



8

Lenguaje JavaScript

JavaScript es un lenguaje de programación de la web que se utiliza para agregar funcionalidad, validar los datos, comunicarse con los servidores web, es decir para añadir contenido dinámico a las páginas web. El código JavaScript va incrustado en las páginas HTML y es el lenguaje de script más utilizado en internet, los navegadores son los encargados de interpretar este código.

Un script es un programa que puede acompañar a un documento HTML o estar contenido en su interior. Las instrucciones del programa se ejecutan cuando se carga el documento, o cuando se produce alguna circunstancia tal como la activación de un enlace o pulsación de un botón por parte del usuario. Este segundo tipo de acciones desencadenan lo que se conoce como eventos.

En esta unidad se estudiará este lenguaje y sus características a través de ejemplos prácticos que mostrarán distintas funcionalidades del mismo.

Contenido

- 8.1 Empezando con JavaScript
- 8.2 Entrada de datos de teclado
- 8.3 Elementos básicos de programación
- 8.4 Estructuras de control
- 8.5 Estructuras repetitivas
- 8.6 Funciones útiles para manejar cadenas de texto
- 8.7 Funciones en JavaScript

Objetivos

- » Conocer las estructuras básicas de programación.
- » Conocer los elementos básicos del lenguaje JavaScript.
- » Realizar *script* sencillos para la entrada y validación de datos.
- » Manejar funciones JavaScript.
- » Realizar *script* sencillos creando funciones.

SABÍAS QUE:

JavaScript es un lenguaje interpretado puro (ni compilación, ni generación de intermedios codificados de ningún tipo) y sensible a mayúsculas y minúsculas, quiere decir que no es lo mismo escribir una variable o una instrucción en mayúscula que en minúscula. Así no es lo mismo escribir alert() que Alert(), alert() es correcto y Alert() es un error.

8.1 Empezando con JavaScript

La integración de JavaScript en documentos XHTML y HTML es muy flexible, ya que existen al menos tres formas para incluir código JavaScript en las páginas web: dentro de la etiqueta <head>, dentro de la etiqueta <body> o en un archivo externo.

Normalmente el código JavaScript se encierra entre etiquetas <script> y se incluye en cualquier parte del documento. Aunque es correcto incluir cualquier bloque de código en cualquier zona de la página, se recomienda definir el código JavaScript dentro de la cabecera del documento; es decir, dentro de la etiqueta <head>.

El formato que utilizaremos para incrustar código JavaScript es:

```
<script type="text/javascript">
<!--
// aquí irá el código
// -->
</script>
```

Los caracteres // se utilizarán para añadir comentarios de línea dentro del código, lo que se escribe dentro no se interpreta, solo se utiliza para comentar. También existen los comentarios de varias líneas que utilizan los caracteres /* para abrir la primera línea del comentario y */ para cerrar la última línea del comentario.

8.1.1 Incluir el script dentro de la etiqueta <head>

En este primer ejemplo, se crea un script en JavaScript que visualiza un mensaje de bienvenida en una alerta y en una línea dentro de la página. Se visualizarán al abrir la página y antes de lo que aparece dentro de la etiqueta <body>.

```
<!DOCTYPE html>
<head>
<title>Ejemplo 1</title>
<script type="text/javascript">
    alert("Este es el mensaje de bienvenida");
    document.write("Este es mi primer ejemplo");
</script>
</head>
<body>
<p>Esto está dentro del body.</p>
</body></html>
```

Como se puede apreciar, el código JavaScript se coloca (o se embebe) dentro de las etiquetas <head> </head>, encerrándolo entre las etiquetas <script></script>. Observa que al final de cada orden de JavaScript se escribe un ; como final de línea.

En este código se utilizan dos órdenes de JavaScript, alert() y document.write():

- alert (): esta orden visualiza una alerta y se muestra como un cuadro de diálogo en la pantalla.

- `document.write()`: se utiliza para escribir en la página. Escribe después de cargar la página.

Para usuarios de IExplore de Windows: cuando se abre una página con código JavaScript incrustado sale en la parte inferior del navegador o en la parte superior (dependiendo de la versión) un mensaje indicando que se ha bloqueado el contenido por encontrar un script y pregunta si se Permite contenido bloqueado. Véase Figura 8.1.



Figura 8.1. Contenido bloqueado con IExplorer.

Hay que decir que con JavaScript se pueden generar, como en muchos otros lenguajes, pequeños programas que vulneran algún “agujero” de seguridad del sistema operativo. En cualquier caso, los ejemplos que veremos aquí no tienen nada de peligrosos. Cuando visites páginas web que no te resulten de confianza el navegador te avisará con la alerta anterior. En ese caso deja las cosas como están; el propio navegador te protegerá de posibles códigos no deseables.

Para este tema pulsaremos en *Permitir contenido bloqueado*.

Una vez creado el código y guardado en un documento HTML, si al abrir con el navegador la página no se visualiza, es probable que haya errores en el código. Para ver los errores de código en el navegador, es necesario configurarlo. Así pues, hacemos lo siguiente:

- Para ver los errores de JavaScript en **IExplorer**. Abrir el navegador menú *Herramientas->Opciones de internet*, pestaña *Opciones avanzadas*, y dentro del apartado *Examinar*, buscar la casilla *Mostrar una notificación sobre cada error de script*, y marcarla (véase Figura 8.2).

RECUERDA:

JavaScript es un lenguaje sensible a mayúsculas y minúsculas, y cada línea de código debe de terminar en ;

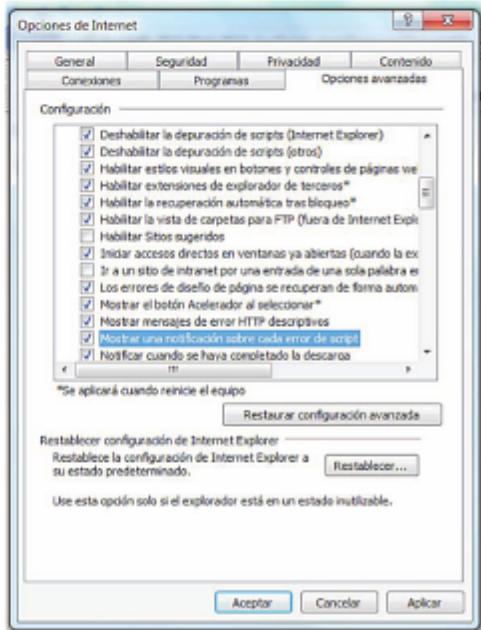


Figura 8.2. Configurar opciones de internet con IExplorer.

SABÍAS QUE:

JavaScript es el único lenguaje de programación que puede ejecutarse en todos los navegadores sin necesidad de cargar plugins adicionales. Es el lenguaje de programación de la web por excelencia en la parte cliente. Se dice que es un *lenguaje de la parte cliente* porque va incluido dentro del código HTML y el navegador es capaz de interpretarlo y ejecutarlo.

Y cuando ocurra un error en la página aparecerá una ventana indicando que hay errores, nos indicará las líneas erróneas y las posiciones donde se encuentra el error (véase Figura 8.3). También se dispone de herramientas de desarrollo (menú Herramientas->Herramientas de desarrollo), desde las que se podrá ver con detalle el código y los errores de las páginas.

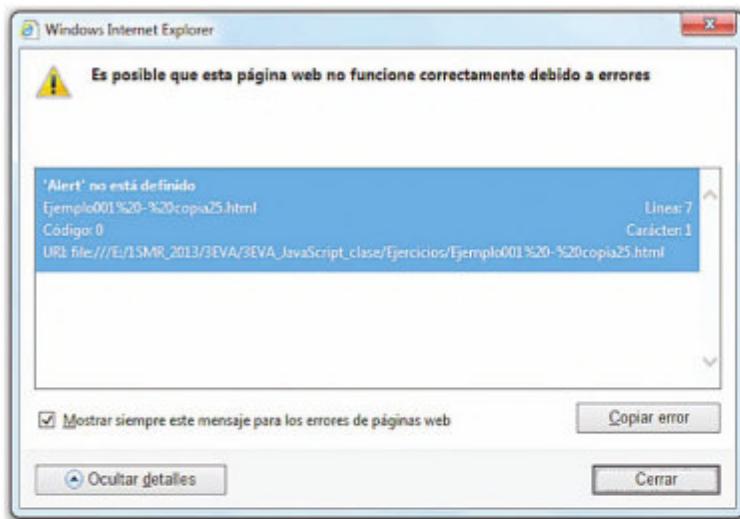


Figura 8.3. Ventana de errores con IEExplorer.

- Para ver los errores de JavaScript en **Firefox**, se abre el navegador, y desde el menú **Herramientas->Consola de error**, o **Desarrollador web ->Consola de errores** (véase Figura 8.4), se abrirán las ventanas de consola. En la ventana de la consola de errores se verán los errores detectados, si se hace clic en el error se abre el documento que nos lleva a la línea errónea.

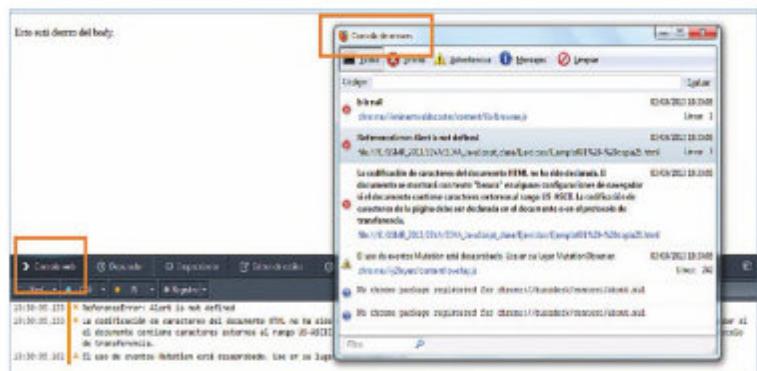


Figura 8.4. Consola web y Consola de errores con Firefox.

- Para ver los errores en **Chrome** pulsamos **Herramientas/Consola Javascript**, se muestra una ventana como la de la Figura 8.5, en la que también se puede ver (pulsando las opciones correspondientes) el código y el error, además de información de la página y el sitio donde se ubica.

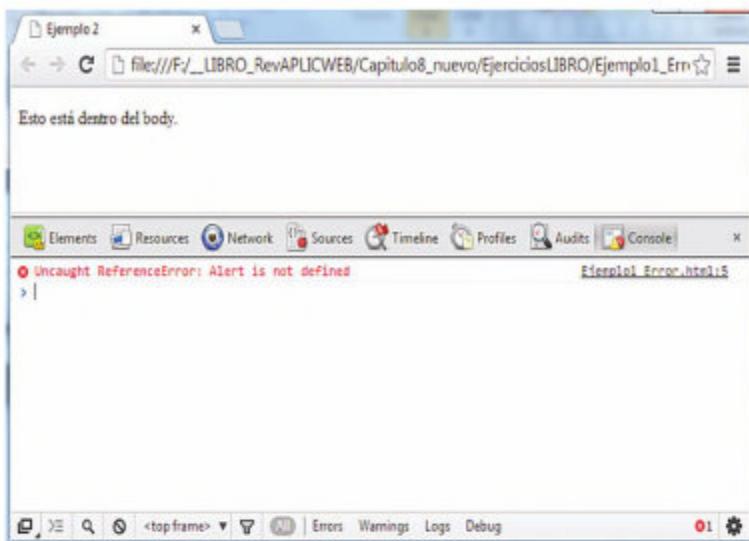


Figura 8.5. Consola JavaScript de Chrome.

8.1.2 Incluir el script en un archivo externo

En este caso se crea un archivo con la extensión .js, en este archivo se escribe el código JavaScript. Al archivo lo llamamos **ejer1.js**, y lo guardamos en una carpeta llamada **js**. El código a escribir es el siguiente:

```
alert("Este es el mensaje de bienvenida.");
document.write("Primeros ejercicios con Javascript.");
```

Lo siguiente es crear la página *Ejemplo001.html* que llama a *ejer1.js*, el código es el siguiente:

```
<!DOCTYPE html>
<head>
  <title>Ejemplo001</title>
  <script type="text/javascript" src="js/ejer1.js"></script>
</head>
<body>
  <p>Esto está dentro del body.</p>
</body>
```

La línea marcada en negrita hace la llamada al archivo externo **ejer1.js** que se encuentra en la carpeta **js**, y lo ejecuta.

8.1.3 Incluir el script dentro del <body> en los elementos HTML o XHTML

Creamos el archivo *Ejemplo0001.html* que incluye código JavaScript en el body. Se incluye de dos maneras, añadiendo un script y añadiendo el evento **onclick** que hace que cuando se pulse en la etiqueta que lo acompaña se ejecute el código añadido, en el ejemplo es una alerta. Teclea este código y prueba la salida.

SABÍAS QUE:

Los navegadores Google Chrome y Mozilla Firefox disponen de un elemento de menú dentro del menú contextual, que va a permitir inspeccionar el código HTML y CSS de las páginas que visitamos. Basita con colocarse en cualquier parte de la página, pulsar el botón derecho del ratón y elegir **Inspeccionar elemento** del menú contextual.

SABÍAS QUE:

Los **lenguajes del lado servidor** son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él. El navegador web hace la función de ser el cliente. Ejemplos de lenguajes del lado servidor son: PHP, PERL, ASP o JSP. Ejemplos del lenguaje del lado cliente son: JavaScript, Applets de Java, VBScript, Flash o CSS.

SABÍAS QUE:



En JavaScript se pueden utilizar indistintamente las comillas " y el apóstrofo ', eso sí, siguiendo el orden, si abres con apóstrofo, debes cerrar con apóstrofo, y si abres con comillas debes cerrar con comillas:

```
document.write("Este es mi primer ejemplo en Javascript");
```

Esta línea hace lo mismo que:

```
document.write('Este es mi primer ejemplo en Javascript');
```

```
<!DOCTYPE html>
<head>
<title>Ejemplo0001</title>
</head>
<body>
<p>Probando Script en el body.</p>
<script type="text/javascript">
    alert("PRUEBA DE JAVASCRIPT");
    document.write("<h2>PRUEBA JAVASCRIPT</h2>");
</script>
<p onclick='alert("Este es el mensaje de bienvenida")'> Haz clic en este párrafo y se visualiza una alerta.</p>
</body>
</html>
```

El mayor inconveniente de este método es que ensucia innecesariamente el código XHTML o HTML de la página y complica el mantenimiento del código JavaScript. En general, este método solo se utiliza para definir algunos eventos y en algunos otros casos especiales.

ACTIVIDAD PROPUESTA 8.1

Añade a los ejercicios anteriores una alerta y un mensaje en la página que visualice tu nombre y tu centro. Utiliza las órdenes `alert()`, y `document.write()`.

8.2 Entrada de datos de teclado

JavaScript, como cualquier otro lenguaje, permite la entrada de datos por teclado, todos los datos tecleados se consideran de tipo texto. En este ejemplo vamos a leer el nombre de una persona y su edad, y lo vamos a visualizar, en una alerta y como un texto normal. Introducimos el concepto de **variable**, que será necesaria para guardar en memoria lo que se lee de teclado.

Para leer los datos de teclado se utiliza la orden **`prompt`**. Pondremos `prompt("mensaje","texto_por_defecto")`, esta orden al ejecutarse visualiza el mensaje en una ventana, y asigna el valor por defecto, si se escribe el `texto_por_defecto`. Este no es obligatorio.

Lo que se lee se debe almacenar en variables para luego trabajar con esa información, por eso se declaran las variables utilizando **`var`**. Crea el documento `Ejemplo2.html` con este código:

```
<!DOCTYPE html>
<head>
<title>Ejemplo 2</title>
<script type="text/javascript">
    var nombre;
```

```

var edad;
// leemos los datos utilizamos la orden prompt
nombre=prompt("Teclea el nombre:");
edad=prompt("Teclea la edad:");
alert("El nombre tecleado es: " + nombre + " y su
edad es: " + edad);
document.write("El nombre tecleado es: " + nombre
+ " y su edad es: " + edad);
</script>
</head>
<body>
<p>Esto está dentro del body.</p>
</body></html>

```

Donde, las instrucciones:

```
var nombre; y var edad;
```

declaran las variables para guardar los datos tecleados en memoria y luego poderlas utilizar.

Las instrucciones:

```

nombre=prompt("Teclea el nombre:");
edad=prompt("Teclea la edad:");

```

hacen posible que aparezca una ventana para introducir los datos que se piden, el dato tecleado se asignará a la variable correspondiente. Al abrir la página en el navegador, aparece una ventana como la que se muestra en la Figura 8.6 para pedir los datos, esto lo hace la orden **prompt**.

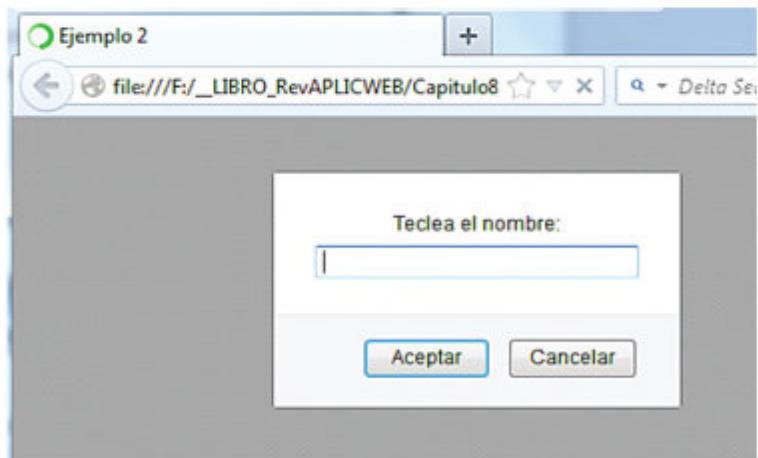


Figura 8.6. Entrada de datos con la orden prompt con Firefox.

Las instrucciones:

```

alert("El nombre tecleado es: " + nombre + " y su
edad es: " + edad);
document.write("El nombre tecleado es: " + nombre + "
y su edad es: " + edad);

```

SABÍAS QUE:

Para ejecutar un programa se necesitan los recursos hardware del ordenador: el procesador (también conocido como UCP, Unidad Central de Proceso, o CPU), la memoria RAM (o memoria principal), los dispositivos de E/S, etc. Las instrucciones de un programa se cargan en la memoria principal y la CPU será la encargada de ejecutarlas.

SABÍAS QUE:

Un lenguaje de programación consta de los siguientes elementos:

- **Un alfabeto o vocabulario (léxico):** formado por el conjunto de símbolos permitidos.
- **Una sintaxis:** son las reglas que indican cómo realizar las construcciones con los símbolos del lenguaje.
- **Una semántica:** son las reglas que determinan el significado de cualquier construcción del lenguaje.

SABÍAS QUE:

Los lenguajes de programación se pueden clasificar atendiendo a varios criterios:

- Según su nivel de abstracción: lenguajes de bajo nivel, de nivel medio y de alto nivel.
- Según la forma de ejecución: lenguajes compilados y lenguajes interpretados.
- Según el paradigma de programación: lenguajes imperativos, funcionales, lógicos, estructurados y orientados a objetos.

visualizan una alerta con los datos tecleados, y una línea en la página con los datos tecleados. Para que aparezcan varios datos los **concatenaremos** utilizando el signo +, los valores fijos los ponemos entre comillas (como “**El nombre tecleado es: ”, o “ y su edad es: ”**) y los valores de las variables con sus nombres (es el caso de nombre y edad). La ejecución visualizará lo que se muestra en la Figura 8.7.

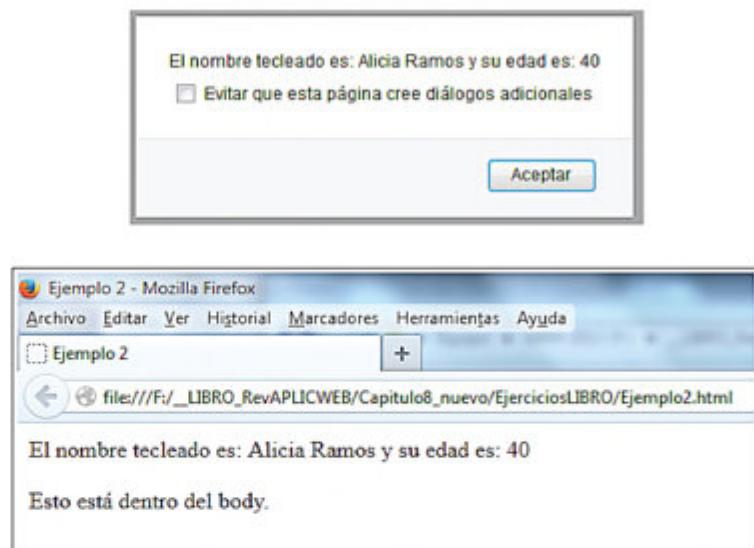


Figura 8.7. Visualización de la página del ejercicio.

También se pueden incluir etiquetas HTML dentro de la orden `document.write`, en este caso se visualizará aplicando las etiquetas HTML. Prueba a cambiar la orden `document.write` en tu ejercicio haciendo que se visualicen los datos con el título `<h2>`:

```
document.write("<h2>El nombre tecleado es: " + nombre +
" y su edad es: " + edad + "</h2>");
```

ACTIVIDAD PROPUESTA 8.2

Escribe un ejercicio con nombre `Ejemplo2_1.html`, que lea el nombre, la edad, la población y el centro de estudios y se visualice en una alerta y dentro de la página:

Me llamo nombre-tecleado, tengo edad-tecleada vivo en población-tecleada, y estudio en centro-tecleado.

Añade al `Ejemplo2_1.html`, que visualice los datos dentro de una tabla, poner borde 1 a la tabla, en la Figura 8.8 se muestra cómo se visualizará la página del ejemplo 2_1:

The figure shows a screenshot of a browser window titled "Ejemplo 2_1.html". The address bar shows "file:///F:/_LIBRO_RevAPLICWEB/Capitulo8_nuevo/EjerciciosLIBRO/Ejemplo2_1.html". The page content displays "El nombre tecleado es: Alicia Ramos y su edad es: 40, la población es: Talavera de la Reina, y el centro es: IES RIBERA DEL TAJO". Below this, there is a table with the following data:

NOMBRE	EDAD	POBLACIÓN	CENTRO DE ESTUDIOS
Alicia Ramos	40	Talavera de la Reina	IES RIBERA DEL TAJO

Below the table, it says "Esto está dentro del body."

Figura 8.8. Visualización de la página del ejemplo 2_1.

8.3 Elementos básicos de programación

Para programar un script es necesario conocer los elementos básicos que se necesitan para construir programas. Así nos encontramos con las variables, los operadores, las estructuras de control, las instrucciones del lenguaje y las funciones. En este apartado vemos las variables y los operadores.

8.3.1 Variables

En realidad, las variables son posiciones de memoria a las que se asigna un nombre y a las que se puede acceder para obtener un valor. Las utilizamos para guardar y para manipular la información.

Las variables se crean normalmente al comienzo del programa utilizando la palabra **var** seguida del nombre de la variable que queremos crear, como en los siguientes ejemplos:

```
var nombre;
var edad;
```

Se pueden declarar dos o más variables en una instrucción. En estos casos se indicará la palabra reservada **var** y, a continuación, la lista de las variables que queremos declarar (los nombres separados por comas):

```
var nombre, edad, poblacion;
```

El nombre o identificador de la variable podrá estar formado por:

- letras (mayúsculas y minúsculas),
- caracteres de subrayado (_),
- el símbolo de dólar (\$).
- y números (del 0 al 9);
- pero **NO** podrá comenzar por un número.
- **NO** podrá coincidir el nombre con alguna de las palabras reservadas del lenguaje (nombres de comandos, etc.).
- Se pueden poner nombres de variables con acentos, pero conviene no ponerlos.

A continuación se muestran algunos ejemplos de identificadores válidos y otros ilegales:

Ejemplos de identificadores válidos	Ejemplos de identificadores ilegales
<code>var vApellido1;</code>	<code>var 1Apellido;</code>
<code>var \$1Apellido;</code>	<code>var #1Apellido;</code>
<code>var _Apellido</code>	<code>var Apellido1°;</code>

Para guardar información en una variable se utilizará el operador de asignación (**=**) precedido por el nombre de la variable y seguido por el valor que queremos asignar. Ejemplos:

```
Nombre = 'Miguel';
nombre = '1111111';
Edad = 17;
```

SABÍAS QUE:

Cuando guardamos información en una variable, cualquier otro valor que dicha variable tuviese anteriormente se perderá (salvo que lo guardemos en otra variable antes de hacer la asignación).

NOTA:

Como JavaScript diferencia mayúsculas y minúsculas en los nombres de variables es muy fácil confundirse; por ello debemos seguir dos normas:

1. Declarar todas las variables al comienzo del programa para poder comprobar rápidamente cualquier duda.
2. Establecer un criterio a la hora de construir identificadores y seguirlo siempre (determinar cuándo se utilizan mayúsculas, minúsculas y subguiones, etc.). A la hora de establecer este criterio deberemos valorar también la legibilidad del programa.

SABÍAS QUE:



Como la visualización de líneas dentro de un `document.write()` y `alert()` sale continua sin saltos de línea, podemos incluir un salto de línea en una orden `document.write()` añadiendo la etiqueta HTML `
` dentro de los datos a visualizar, por ejemplo:

```
document.write("<br />Nombre: " + nombre + "  
<br />Edad: " + edad);
```

Y si queremos incluir saltos de línea (`\n`), o tabulador (`\t`) en una alerta, utilizaremos los caracteres que vienen entre paréntesis. Ejemplo:

```
alert("Nombre: " + nombre  
+ "\nEdad: " + edad);  
  
alert(" \nNombre: "  
+ nombre + "\t y  
edad : " + edad);
```

En el siguiente ejemplo (`ejemplo3.html`) se declaran dos variables, nombre y edad, se leen de teclado, se visualizan en el documento, se asignan nuevos valores a esas variables, y de nuevo se visualizan en el documento. Así observamos la variación del contenido de las variables:

```
<!DOCTYPE html >  
<head>  
<title>Ejemplo 3</title>  
<script type="text/javascript">  
    var nombre, edad;  
    // leemos los datos utilizamos la orden prompt  
    nombre=prompt("Teclea el nombre: ");  
    edad=prompt("Teclea la edad: ");  
    // Visualizamos los datos leídos  
    document.write("El nombre tecleado es: " + nombre +  
    " y la edad es: " + edad);  
    // Asignamos nuevos valores a esas variables  
    nombre = "NOMBRE NUEVO";  
    edad = 555;  
    // Visualizamos de nuevo y veremos que han cambiado  
    document.write("El nombre nuevo es: " + nombre + "  
y la edad nueva es: " + edad);  
</script>  
</head>  
<body>  
<p>Esto está dentro del body.</p>  
</body></html>
```

ACTIVIDAD PROPUESTA 8.3

Escribe un ejercicio con nombre `Ejemplo3_1.html`, que lea de teclado la población, la provincia y la calle y visualice en la pantalla los datos leídos, cada uno en una línea. Visualizarlo también en una alerta.

8.3.2 Operadores y expresiones

Los **operadores** sirven para manejar o transformar la información, para realizar cálculos y para comparar información. Las **expresiones** son combinaciones de variables, constantes y operadores que devuelven un valor realizando determinadas operaciones. Así pues, en una expresión nos encontramos:

- **Operandos** son los elementos cuyo valor se manipula u opera, serán variables, constantes o expresiones.
- **Operador** u operadores son los elementos que determinan el tipo de operación a realizar:

Ejemplo de expresiones son:

Expresión	Operandos	Operadores
Suma = A + B	Suma, A, B	= , +
A >= B	A, B	>=
RESUL = A * 20	RESUL, A, 20	= , *
(A==B && C>D);	A, B, C, D	== , && , >

Nombre de operador	Operador	Ejemplo de expresiones
De asignación , asigna un valor.	=	Numero = 10;
De concatenación , une cadenas de caracteres.	+	"El nombre nuevo es: " + nombre nombre + apellido
Aritméticos , se utilizan para cálculos aritméticos u operaciones matemáticas.	Suma: + Resta: - Multiplicación: * División: / División entera: \ Resto de la división: %	total = importe + iva Salario = base - retenciones importe = precio * unidades resultado1 = cantidad / 5 resultado2 = cantidad \ resto = cantidad % 5
De comparación , se utilizan para comparar valores, y devuelven true o false, dependiendo de si se cumple o no la comparación. Se utilizan en programación para toma de decisiones.	Igual: == Distinto: <> Mayor que: > Menor que: < Mayor o igual: >= Menor o igual: <=	(n1 == n2), compara si n1 y n2 son iguales. (n1 <> n2), compara si n1 y n2 son distintos. (n1 > n2), compara si n1 es menor que n2. (n1 < n2), compara si n1 es mayor que n2. (n1 >= n2), compara si n1 es mayor o igual que n2. (n1 <= n2), compara si n1 es menor o igual que n2.
Operadores lógicos , se utilizan para unir varias expresiones de comparación.	Y (AND): && O (OR): NO (NOT): !	(n1 == n2) && (A>B), evalúa si n1 es igual a n2 y A es mayor que B, devolverá true si se cumplen las dos condiciones. (n1 > n2) (n1>5), evalúa si n1 es mayor que n2, o mayor que 5. Si se cumple cualquiera de las dos condiciones, devolverá true. !(A>B) Devuelve true cuando el operando es falso. Es decir devuelve true cuando NO es A mayor que B.

En el siguiente ejemplo (*Ejemplo4.html*) se declaran tres variables, n1 y n2, que se leen de teclado, y suma que almacenará la suma de los dos números. Una vez leídas n1 y n2, se calcula la suma y se visualiza el resultado en una alerta y en el documento:

IMPORTANTE: Para leer datos numéricos enteros utilizaremos la orden **parseInt()**:

```
parseInt(prompt("Mensaje a visualizar",
valor_por_defecto));
```

Esta orden convierte el dato tipo texto a numérico. Si no lo ponemos así, en la suma de los números aparecerá como resultado la concatenación de los números tecleados. En el *Ejemplo4* escribiremos:

```
n1=parseInt(prompt("Teclea el numero 1:",0));
n2=parseInt (prompt("Teclea el numero 2:",0));
```

```
<!DOCTYPE html>
<head>
<title>Ejemplo 4</title>
<script type="text/javascript">
// declaramos las variables
var n1, n2, suma;
// Leemos de teclado n1 y n2
n1=parseInt(prompt("Teclea el número 1:",0));
n2=parseInt (prompt("Teclea el número 2:",0));
```

SABÍAS QUE:

No todas las expresiones requieren de la utilización de operadores, en realidad tan solo se requiere que devuelva un valor. En este sentido podemos considerar que una constante o una variable constituyen por sí solas una expresión.

En las expresiones pueden intervenir otros elementos, como son las funciones.

SABÍAS QUE:

El lenguaje de más bajo nivel por excelencia es el lenguaje máquina que es entendible directamente por la máquina. Las instrucciones están formadas por cadenas de ceros y unos, es decir, utilizan el alfabeto binario (0 y 1). Los programas en este lenguaje son específicos para cada procesador.

```
// Calculamos la suma
suma=n1+n2;
// Visualizamos la suma
document.write("<br />La suma es: " + suma);
alert("La suma es: " + suma );
</script>
</head>
<body>
<p>Esto está dentro del body.</p>
</body> </html>
```

El siguiente código calcula el cuadrado y el cubo del número 4. Se declaran tres variables: *numero*, *cuadrado* y *cubo*. Se asignará a *numero* el valor 4 en la declaración. Se realizan los cálculos y el resultado se visualiza en una alerta y en el documento. Crea la página *Ejemplo5.html* que incluya este script.

```
<script type="text/javascript">
    // declaramos variables
    var numero=4, cuadrado, cubo;
    // Calculamos
    cuadrado=numero*numero;
    cubo=cuadrado*numero;
    // Visualizamos resultados
    document.write("<br />El cuadrado es : " + cuadrado
    + "<br />El cubo es : " + cubo);
    alert("El cuadrado es : " + cuadrado + "\nEl cubo
    es : " + cubo);
</script>
```

ACTIVIDAD PROPUESTA 8.4

Cambiar el ejercicio anterior y llamarle *Ejemplo5_1.html*. En este caso, leer el número de teclado y calcular su cuadrado y su cubo.

Cambiar el ejercicio anterior y llamarle *Ejemplo5_2.html*. En este caso, leer dos números de teclado y calcular la suma, la resta del primero menos el segundo, la multiplicación, la división del primero por el segundo, y el resto de dividir el primero por el segundo. Visualizar el resultado en una alerta y en el documento (véase Figura 8.9).

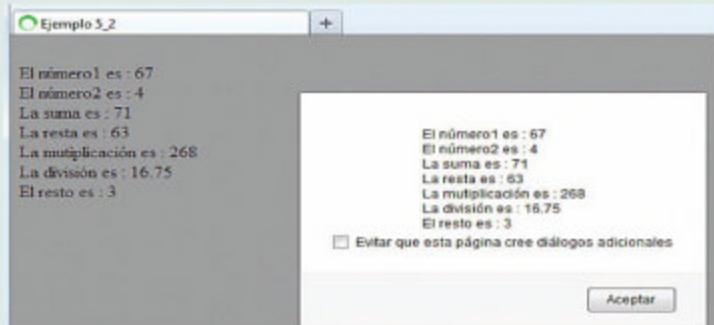


Figura 8.9. Visualización de la página del *Ejemplo5_2*.

En el siguiente ejemplo (*Ejemplo6.html*) se visualiza en el documento y en una alerta la comparación entre dos variables con valores asignados. Las expresiones de comparación devuelven un valor lógico true o false y al escribirlos se incluyen entre paréntesis, por ejemplo se pondrá (A==B) para comparar si A es igual a B, o (A>B) para comparar si A es mayor que B.

```
<script type="text/javascript">
    var A = 10, B = 12;
    document.write("<br/>A= " + A + " y B= " + B);
    document.write("<br/>A==B es: " + (A==B) + "<br/>A>B
es: " + (A>B));
    alert ("<br/>A= " + A + " y B= " + B + "<br/>A==B es: " +
(A==B) + "<br/>A>B es: " + (A>B));
</script>
```

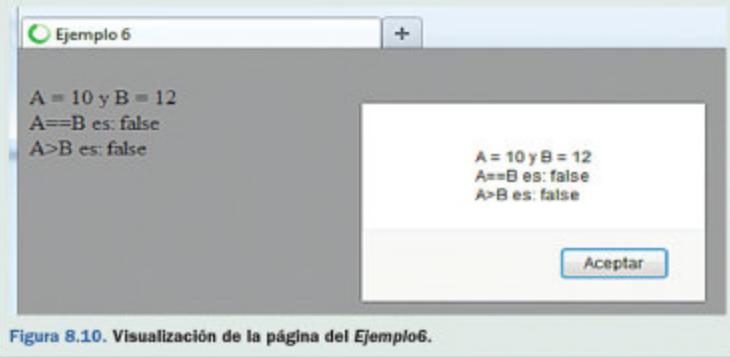


Figura 8.10. Visualización de la página del Ejemplo6.

ACTIVIDAD PROPUESTA 8.5

A partir del ejercicio anterior, añade las líneas necesarias para visualizar las comparaciones de A menor que B.

Guarda con el nombre *Ejemplo6_1*.

Cambia el ejercicio anterior, guárdale con el nombre *Ejemplo6_2*. Ahora en lugar de asignar valores a A y a B, haz que se lean de teclado y se muestre en la página y en una alerta el resultado de comparar si son iguales, si la suma de los dos números es mayor de 100, y si la resta de A – B es menor de 100. Antes de comparar hay que sumar los números y guardarlos en una variable, y restarlos y guardarlos en otra variable. Pon títulos de nivel <h3> para que la página se muestre como en la Figura 8.11.

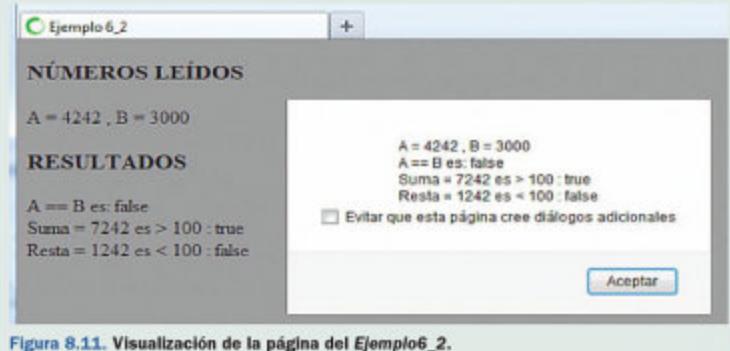


Figura 8.11. Visualización de la página del Ejemplo6_2.

SABÍAS QUE:

Dentro de la orden `document.write`, en el mensaje, para que se visualice el signo < lo escribiremos entre espacios, ya que el navegador puede pensar que es una etiqueta HTML; es decir, este `A<B`, no se visualizaría; pero si lo ponemos con un espacio `A < B`, sí se visualizará. Va a ocurrir siempre que pongamos el carácter <, con lo cual conviene añadir uno o dos espacios.



SABÍAS QUE:

Para representar el funcionamiento de los operadores lógicos AND, OR y NOT se utilizan Tablas de verdad, en las que se coloca la variable con su valor (True o False), y el resultado de operar (True o False), lo escribimos así:

A	B	A AND B
True	True	True
True	False	False
False	True	False
False	False	False

A	B	A OR B
True	True	True
True	False	True
False	True	True
False	False	False

A	NOT A
True	False
False	True

Con los operadores lógicos se pueden realizar comparaciones complejas. El resultado de cualquier operación que utilice operadores lógicos siempre es un valor lógico o booleano: true o false.

El funcionamiento de estos operadores es el siguiente:

- La negación lógica NOT (!). La comparación devuelve true si la variable es false, y false si la variable es true. Hay que tener en cuenta lo siguiente:
 - Una variable es lógica si se la asigna true o false. Ejemplo: var lógica = true;
 - Si la variable contiene un número, si vale 0 la transformación lógica es false y será true para cualquier otro número (positivo o negativo, decimal o entero).
 - Si la variable contiene una cadena de texto, la transformación lógica es false si la cadena es vacía ("") , y es true en cualquier otro caso.

Ejemplos:

Valor de la variable	Comparación	Resultado de la comparación
A = true;	!A	false
Numel=0;	!Numel	false
B=false;	!B	true
Nume2=100;	!Nume2	true
Cadenal="HOLA";	!Cadenal	true
Cadena2=""	!Cadena2	false

- La operación lógica AND (&&) obtiene su resultado combinando dos valores booleanos. El resultado solamente es true si los dos operandos son true:

Valor de la variable	Comparación	Resultado de la comparación
A=true; B=false;	A && B	false
A=true; B=true;	A && B	true
A=false; B=false;	A && B	false
A=false; B=true;	A && B	false

- La operación lógica OR (||) obtiene su resultado combinando dos valores booleanos. El resultado es true si algún operando es true, solamente es falso si los dos operandos son falsos:

Valor de la variable	Comparación	Resultado de la comparación
A=true; B=false;	A B	true
A=true; B=true;	A B	true
A=false; B=false;	A B	false
A=false; B=true;	A B	true

En el siguiente código se realizan comparaciones complejas en las que los operadores lógicos AND y OR unen varias expresiones. Por ejemplo, a partir de las variables A, B, C y D podemos evaluar si A es igual a B y C es igual a D, y se escribirá ($A==B \&& C==D$). O también podremos preguntar si D es mayor que A o B es igual a C, ($D>A || B==C$). Copia y prueba el siguiente código, guárdalo como *Ejemplo7.html*.

```
<script type="text/javascript">
var A = 10, B = 20, C = 30, D = 40;
document.write("<h3>COMPARACIONES COMPLEJAS</h3>");
document.write("A=" + A + " B=" + B + " C=" + C + " D=" + D);
document.write("<br /> =====");
document.write("<br />A==B && C==D: " + (A==B && C==D));
document.write("<br />A>B && C>D: " + (A>B && C>D));
document.write("<br />A< B && C< D: " + (A< B && C<D));
document.write("<br /> =====");
document.write("<br />A==B || C==D: " + (A==B || C==D));
document.write("<br />D>A || C>D: " + (D>A || C>D));
document.write("<br />A< B || C> D: " + (A< B || C> D));
document.write("<br /> =====");
document.write("<br /> !(A==B): " + !(A==B));
document.write("<br /> !(D>A): " + !(D>A));
document.write("<br /> !(A< B): " + !(A< B));
</script>
```

ACTIVIDAD PROPUESTA 8.6

Cambia el ejercicio anterior y muestra las comparaciones de evaluar si C y D son distintos. Si A no es mayor que B. Si C es mayor que A y que B. Si D es menor que A y que B. Si A es menor que B o A es mayor que D. Si A es igual a B o A igual a C. Utiliza los operadores lógicos. Guárdalo con el nombre *Ejemplo7_1*.

8.4 Estructuras de control

Las pruebas realizadas hasta ahora han consistido en la ejecución lineal de una serie de instrucciones básicas. En este apartado vamos a incluir las estructuras de control de flujo que van a permitir realizar una serie de instrucciones o realizar otra serie de instrucciones dependiendo de si se cumple o no una condición. Es decir, vamos a poder preguntar por el valor de las variables o expresiones y dependiendo del resultado se ejecutarán unas acciones o se ejecutarán otras.

8.4.1 Estructura *if*

Se emplea para tomar decisiones en función de una condición. Si se cumple una determinada condición se ejecutará un conjunto de instrucciones, si no se cumple no se hace nada, sigue en secuencia. El diagrama de flujo se representa en la Figura 8.12 y el formato es el siguiente:

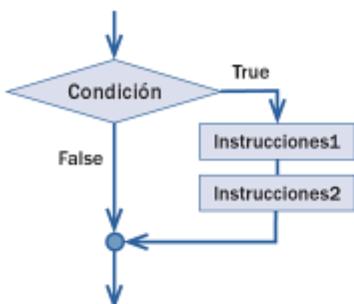


Figura 8.12. Estructura if.

```

if (condición)
{
    // se añaden las instrucciones a ejecutar
    // si se cumple la condición
    Instrucciones1;
    Instrucciones2;
}
  
```

Donde:

- *condición* es cualquier expresión que devuelva un valor lógico (*true/false*). Si la condición se cumple (es decir, si su valor es *true*) se ejecutan todas las instrucciones que se encuentran dentro de *{...}*. Si la condición no se cumple (es decir, si su valor es *false*) no se ejecuta ninguna instrucción contenida en *{...}* y el programa continúa ejecutando el resto de instrucciones del script.
- *Instrucciones* es un bloque de una o más instrucciones que se ejecutarán solo en el caso de que se cumpla la condición. Si se trata de una sola instrucción podemos prescindir de las llaves *({})*.

El código del siguiente script (*Ejemplo8.html*) visualiza la alerta con la variable *Mensaje*, cuyo contenido es: "Probando la estructura if" porque la variable *mostrarMensaje* es igual a *true*. Prueba el mismo ejemplo asignando *false* a la variable *mostrarMensaje*.

```

<script type="text/javascript">
var Mensaje= "Probando la estructura if";
var mostrarMensaje = true;
if(mostrarMensaje)
{
    alert(Mensaje);
}
</script>
  
```

El ejemplo se podría reescribir también como:

```

var Mensaje= "Probando la estructura if";
var mostrarMensaje = true;
if(mostrarMensaje==true)
{
    alert(Mensaje);
}
  
```

El siguiente ejemplo (*Ejemplo09.html*) lee una nota introducida por teclado y, si la nota es igual o mayor que 5, visualizará una alerta con el texto */APROBADO!*, en caso contrario no hará nada.

```

<script type="text/javascript">
var Nota;
Nota = prompt("Introduce la nota del alumno: ", 0);
  
```

```
if (Nota >= 5)
{
    alert("¡APROBADO!");
}
</script>
```

ACTIVIDAD PROPUESTA 8.7

Lee dos notas de teclado que sean datos numéricos. Calcula la media de las dos notas y visualiza en una alerta y en el documento las dos notas, la nota media y si está aprobado. Pon títulos al visualizar en el documento. Guárdalo como Ejemplo09_1.html.

8.4.2 Estructura if...else

En este caso se realizarán unas instrucciones si se cumple la condición, y otras instrucciones si no se cumple. La estructura se llama *if...then...else...*; es decir *si condición entonces unas instrucciones si no otras instrucciones*. El diagrama de flujo se representa en la Figura 8.13 y el formato es el siguiente:

```
if (condición)
{ // si se cumple la condición
    Instrucciones1; Instrucciones2;
    Instrucciones3;
}
else
{ // si no se cumple la condición
    Instrucciones4; Instrucciones5;
};
```

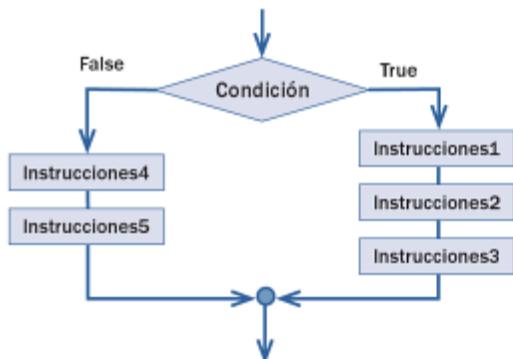


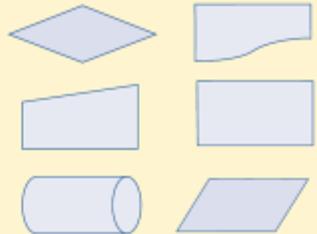
Figura 8.13. Estructura if...else

Si la condición se cumple (es decir, si su valor es true) se ejecutan todas las instrucciones que se encuentran dentro del `if()`. Si la condición no se cumple (es decir, si su valor es false) se ejecutan todas las instrucciones contenidas en `else { }`.

El siguiente código lee de teclado la edad de un alumno y si el valor de la variable `edad` es mayor o igual que el valor numérico 18, la condición del `if()`

SABÍAS QUE: ?

Para representar el diseño de un programa se emplean algunas herramientas gráficas como los diagramas de flujo o el pseudocódigo. Algunos símbolos son:



SABÍAS QUE: ?

Las construcciones fundamentales para la **programación estructurada** son:

- La **secuencial**: implementa los pasos del proceso esenciales para la especificación de cualquier algoritmo.
- La **condicional**: proporciona las funciones para procesos seleccionados a partir de una condición lógica (que se representa mediante un rombo). Si se cumple la condición se realiza la tarea de la *Parte Sí*, si no se cumple se realiza la tarea de la *Parte No*. La selección múltiple es una extensión de la anterior.
- La **repetitiva**: proporciona los bucles. *Repetir-hasta*: primero ejecuta la tarea del bucle y después comproueba la condición, si no se cumple se vuelve a realizar la tarea; si la condición se cumple finaliza el bucle, la tarea se realiza al menos una vez. *Hacer-mientras*: primero se comprueba la condición y después se realiza la tarea del bucle repetidamente siempre y cuando la condición se cumpla; el bucle finaliza cuando la condición no se cumple.

se cumple y, por tanto, se ejecutan sus instrucciones y se muestra el mensaje "Eres mayor de edad". Sin embargo, cuando el valor de la variable *edad* no es igual o mayor que 18, la condición del *if()* no se cumple, por lo que automáticamente se ejecutan todas las instrucciones del bloque *else { }*. En este caso, se mostraría la alerta "Todavía eres menor de edad".

```
<script type="text/javascript">
var edad = prompt("Teclea la edad: ", 0);
if(edad >= 18) {
    alert("Eres mayor de edad.");
}
else {
    alert("Todavía eres menor de edad.");
}
</script>
```

ACTIVIDAD PROPUESTA 8.8

Modifica la Actividad propuesta 8.7, indicando si la media del alumno es mayor o igual a 5 que visualice APROBADO, y si no lo es que visualice SUSPENSO. Guárdalo como *Ejemplo9_2.html*.

Realiza una página que lea el nombre, la edad y la nota de un alumno y visualice en el documento (utiliza *document.write*) el nombre del alumno, su edad, si es mayor de edad o menor de edad, su nota y si está aprobado o suspenso. Utiliza títulos. Guárdalo como *Ejemplo9_3.html*.

Todas las estructuras de control se pueden anidar, es decir escribir una dentro de otra. En el siguiente ejemplo (*Ejemplo10.html*) se utiliza el anidamiento de estructuras para realizar comprobaciones adicionales.

```
<!DOCTYPE html>
<head>
<title>Ejemplo 10</title>
<script type="text/javascript">
// Ejemplo de anidamiento de estructuras alternativas.
var Nota;
Nota = prompt("Introduce la nota del alumno entre 5 y 10: ", 0);
if (Nota >= 5 && Nota <= 10)
{
    alert("¡APROBADO!");
}
else
{
    if (Nota >= 0 && Nota < 5)
    {
        alert("¡suspenso!");
    }
    else
    {
        alert("Nota errónea");
    }
}
```

```
</script>
</head>
<body>
<p>Esto está dentro del body.</p>
</body></html>
```

ACTIVIDAD PROPUESTA 8.9

- Crea una página que incluya un script que lea (mediante `prompt`) una cantidad introducida por teclado, y si es 100 que visualice en el documento la cantidad y la cantidad más 200. Si la cantidad está entre 101 y 200 que visualice en el documento la cantidad y la cantidad multiplicada por 2. Si la cantidad es negativa que visualice en el documento la cantidad y ES NEGATIVA, en otro caso que visualice en el documento la cantidad y la cantidad multiplicada por 3. Guárdalo como *Ejemplo10_1.html*.
- Crea una página que incluya un script que lea (mediante `prompt`) una edad introducida por teclado y visualice que no es correcta la edad si es mayor de 110, o es menor de 0. Visualice que es mayor de edad si es mayor de 18, y visualice menor de edad si es menor de 18. Utiliza la orden `document.write`. Guárdalo como *Ejemplo10_2.html*.
- Crea una página que incluya un script para introducir por teclado el sexo, que deberá ser H o M o h o m y visualizar en una alerta Hombre si es H o h, Mujer si es M o m y error si no es H, h, m o M. Guárdalo como *Ejemplo10_3.html*.
- Crea una página que incluya un script para leer de teclado dos números y visualizar si son iguales, o cuál es el mayor y cuál el menor. Guárdalo como *Ejemplo10_4.html*.



SABÍAS QUE:

Un programa que se escribe en un lenguaje de alto nivel tiene que traducirse a un código que pueda utilizar la máquina. Los programas traductores que pueden realizar esta operación se llaman **compiladores** o **intérpretes**.



RECUERDA:

Para leer datos numéricos enteros utilizaremos la orden `parseInt()`.

8.4.3 Estructura alternativa múltiple

Esta opción se utiliza cuando se quiere comprobar los múltiples valores que puede tomar una variable o una expresión, puede sustituir a muchos `if` encadenados. El diagrama de flujo se muestra en la Figura 8.14 y el formato es el siguiente:

```
switch (expresión)
{
    case valor1:
        instrucciones1;
        break;
    case valor2:
        instrucciones2;
        break;
    case valor3:
        instrucciones3;
        break;
    ...
    case valorn:
        instruccionesn;
        break;
    [default:
        instrucciones;]
}
```

SABÍAS QUE:

Un **compilador** es un programa que pude de leer otro programa escrito en un determinado lenguaje (un lenguaje fuente) y traducirlo en un programa equivalente en otro lenguaje (lenguaje destino). El compilador devolverá errores si el programa en el lenguaje fuente no está bien escrito. El programa destino se podrá ejecutar si el lenguaje destino es directamente ejecutable por la máquina.

Donde:

- **expresión** es cualquier expresión o variable válida con la que se compararán los valores que acompañan a la cláusula `case`.
- **valor1..valorn** son valores que suponemos puede tomar la expresión o variable a los que les sigue la instrucción o instrucciones que queremos que se ejecuten en cada caso.
- **default** es una cláusula opcional, se ejecutarán las instrucciones que la siguen en el caso de que el valor no coincidiese con ninguno de los casos contemplados.

El funcionamiento de esta estructura es el siguiente:

1. Calcula el valor de expresión.
2. Comprueba desde el principio cada valor que acompaña a las cláusulas `case` hasta encontrar alguno que coincida con el valor de expresión.
3. Cuando encuentra un valor que coincide con `expresión` ejecuta las instrucciones correspondientes hasta que encuentra la cláusula `break`. Los valores si no se pasan a numéricos (por ejemplo, al leer, si se pone `parseInt` aseguramos que la entrada es un número entero) se han de preguntar entre comillas, como caracteres o cadenas, porque los considera caracteres.
4. Si no encuentra ningún valor que coincida con `expresión` ejecuta las instrucciones correspondientes a la cláusula `default` (si existe, en caso contrario no hará nada).
5. Sale del bloque de la estructura `switch..case`.

Observaciones:

- La cláusula `break` que aparece en este formato no es obligatoria pero si no se utiliza cambiará el funcionamiento de la estructura `switch ..case`. Esta cláusula es la responsable la salida del bloque una vez que se ejecuten las instrucciones de una de las cláusulas `case`. Si no se pone, se seguirán ejecutando todas las instrucciones hasta llegar al final. Es decir, las cláusulas `case` sin `break` actuarían como puntos de entrada a la estructura, siendo la salida la misma en todos los casos.
- Para una misma cláusula `case` se pueden especificar diversos valores en lugar de un único valor, en este caso se indicarán los valores separados por comas según el formato:

```
case valor1:  
case valor2:  
case valor3: instrucciones;  
              break;
```

Por ejemplo, se visualiza una alerta si la expresión que se evalúa vale 10, 20 o 30:

```
case 10:  
case 20:  
case 30: alert("El valor de la variable es 10, 20 o  
            30");  
          break;
```

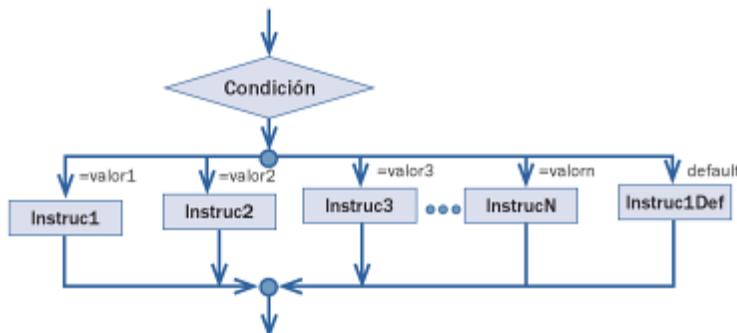


Figura 8.14. Estructura switch.

El siguiente ejemplo (*Ejemplo11.html*) muestra la utilización de una estructura alternativa múltiple anidada dentro de una estructura alternativa doble.

```

<!DOCTYPE html>
<head>
<title>Ejemplo 11</title>
<script type="text/javascript">
// Ejemplo de estructura alternativa múltiple
// (Anidada dentro de una alternativa doble).
var Nota, Calificacion = " ";
Nota = parseInt(prompt("Introduce la nota del alumno: ", 0));
Nota = Math.round(Nota); // redondea la nota a un entero
if (Nota >= 0 && Nota <= 10)
{
    switch(Nota)
    {
        case 5:
            Calificacion = "Suficiente";
            break;
        case 6:
            Calificacion = "Bien";
            break;
        case 7:
        case 8:
            Calificacion = "Notable";
            break;
        case 9:
        case 10:
            Calificacion = "Sobresaliente";
            break;
        default:
            Calificacion = "Suspensos";
    };
}
else
{
    Calificacion = "Nota errónea";
}
  
```

SABÍAS QUE:

Los **intérpretes** son otra alternativa para traducir los programas escritos en lenguaje de alto nivel. En este caso, en vez de producir un programa destino como resultado del proceso de traducción, el intérprete nos da la apariencia de ejecutar directamente las operaciones especificadas en el programa fuente con las entradas proporcionadas por él. Cada vez que se ejecuta una instrucción se debe interpretar y traducir a lenguaje máquina.

SABÍAS QUE:



El código de un programa pasa por diferentes estados desde que se escribe hasta que se ejecuta en el ordenador: código fuente, código objeto y código ejecutable.

```
alert(Calificacion);
</script>
</head>
<body>
<p>Esto está dentro del body.</p>
</body></html>
```

En esta ocasión, en lugar de ejecutar una alerta para cada caso, se ha optado por guardar el texto a mostrar en la alerta en la variable *Calificación* para, al final, mostrar el contenido de dicha variable. Observa también que la variable *Nota* se redondea antes de comprobar su contenido (*Nota = Math.round(Nota);*) ya que los valores de las cláusulas *case* deben ser exactos (5, 6, etc.).

El siguiente script (*Ejemplo12.html*) lee el número del día de la semana y visualiza en una alerta el nombre del día, el 1 es lunes, el 2 es martes, el 3 es miércoles, el 4 es jueves, y así sucesivamente.

```
<script type="text/javascript">
var dia, nombre;
dia = parseInt(prompt("Teclea el día de la semana: ",
0));
switch(dia)
{case 1: nombre = "Es lunes"; break;
case 2: nombre = "Es martes"; break;
case 3: nombre = "Es miércoles"; break;
case 4: nombre = "Es jueves"; break;
    case 5: nombre = "Es viernes"; break;
case 6: nombre = "Es sábado"; break;
case 7: nombre = "Es domingo"; break;
    default: nombre = "Día erróneo.";}
alert(nombre);
</script>
```

SABÍAS QUE:



El código fuente es el código escrito por los programadores utilizando algún editor de texto o alguna herramienta de programación. Se utiliza un lenguaje de programación de alto nivel apropiado para el problema que se trata de resolver. Para escribir el código se parte de los diagramas de flujo o pseudocódigos diseñados en la etapa de diseño. Este código no es directamente ejecutable por el ordenador.

ACTIVIDAD PROPUESTA 8.10

- Crea una página que incluya un script que permita leer un mes de teclado (el número de mes), y visualice en una alerta el nombre del mes leído: enero, febrero, marzo, etc. Guarda el ejercicio como *Ejemplo12_1.html*.
- Crea una página que incluya un script que permita leer dos números de teclado y un carácter que será la operación de cálculo a realizar. Este carácter será * para multiplicar, + para sumar, - para restar y / para dividir. Utiliza una estructura *switch* para preguntar por la operación y realiza el cálculo. Visualiza el resultado de la operación en el documento de manera que se vean los dos números, la operación y el resultado. Por ejemplo, si leemos de teclado 10, 30 y +, se ha de visualizar en la página: $10 + 30 = 40$. Si la operación tecleada es errónea se visualiza en el documento el mensaje "Error en la operación."

Guarda el ejercicio como *Ejemplo12_2.html*.

- Crea una página que incluya un script que permita leer un número de mes de teclado y a partir de ese mes calcula la productividad. La productividad es $100 + \text{el factor del mes}$. Y el factor del mes es el siguiente:
 Los meses enero, febrero y marzo tienen factor 15.
 Los meses abril, mayo y junio tienen factor 17.
 Los meses septiembre, octubre y noviembre tienen factor 20.
 Y diciembre tiene factor 21.
 Si no son esos meses el factor es 0.
 Visualiza la productividad en el documento, indicando:
 El nombre de mes es Nombre-Mes y la productividad resultado.
 Visualiza "Error en el mes" si el número de mes tecleado no está entre 1 y 12.
 Guarda el ejercicio como *Ejemplo12_3.html*.
- Crea una página que incluya un script para leer de teclado el nombre de empleado, su salario y su número de departamento. Se pretende subir el salario de los empleados. Para subir el salario preguntamos por el número de departamento.
 Si el departamento es 10, la subida de salario será de 100.
 Si el departamento es 20, la subida de salario será del 5 %.
 Si el departamento es 30, la subida de salario será del 4 %.
 Si es otro departamento la subida será de 75.
 Visualiza en el documento el nombre del empleado, el salario, la subida de salario, y el nuevo salario que será la suma del salario más la subida.
 Visualiza:
Nombre empleado: nombre_leido, su salario es: salario_leido, su departamento es: depart_leido, la subida es: subida_calculada, y el nuevo salario es: salario_leido más la subida_calculada.
 Guarda el ejercicio como *Ejemplo12_4.html*.

8.5 Estructuras repetitivas

En ocasiones necesitaremos que un bloque de instrucciones se ejecute varias veces seguidas; en estos casos utilizaremos estructuras repetitivas o bucles. En JavaScript disponemos de las siguientes:

8.5.1 Estructuras while

La estructura *while* ejecuta un bloque de instrucciones y repite dicha ejecución mientras que se cumpla una condición. El diagrama de flujo se representa como se muestra en la Figura 8.15 y el formato es el siguiente:

```
while(condición)
{
  Instrucciones;
}
```

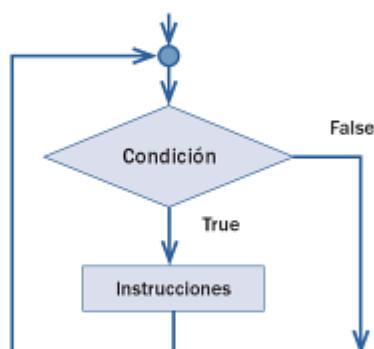


Figura 8.15. Estructura while.

Donde:

- *condición* es la condición a evaluar, si su valor es true se entrará en el bucle y se repetirá mientras se cumpla la condición. Será comprobado antes de cada nueva ejecución del bloque de instrucciones.
- *instrucciones* es el bloque de instrucciones que se ejecutará si se cumple la condición.

Funcionamiento:

1. Al encontrar la estructura `while` lo primero que hace (antes de entrar por primera vez en el bucle) es evaluar la condición: si es verdadera entra en el bucle y ejecuta el bloque de instrucciones, pero si es falsa ni siquiera llegará a entrar en el bucle.
2. Una vez ejecutadas las instrucciones del bucle se evalúa de nuevo la condición para determinar si se vuelve a ejecutar el bloque o no (si es verdadera se ejecuta, si es falsa deja de ejecutarse). Este punto se repite hasta que la condición deja de ser verdadera.
3. Cuando al evaluar la condición el resultado es *false*, el flujo del programa va a la línea siguiente al final del bucle.

El siguiente ejemplo (*Ejemplo13.html*) utiliza esta estructura para validar la lectura de una nota, lo que hace es leer una nota que esté comprendida entre 0 y 10 y así asegurarse que se teclea una nota correcta. Primero se lee la nota y el bucle verifica que sea correcta. La última instrucción del bucle es volver a leer la nota. Se saldrá del bucle cuando se teclee la nota correcta. El `while` se ejecuta mientras la nota sea menor de 0 (error, nota negativa) o la nota sea mayor de 10 (nota errónea), en ese caso se lee.

```
<script type="text/javascript">
// Ejemplo de estructura while
var Nota;
// Leemos la nota antes de entrar en el bucle
Nota = prompt("Introduce la nota del alumno: ", 0);
while (Nota < 0 || Nota > 10)
{
    // Si se entra en el bucle es porque la nota no es correcta
    // Con lo cual hay que leer de nuevo hasta que sea correcta.
    alert("Nota errónea: "+Nota+". Teclea de nuevo.");
    Nota = prompt("Introduce la nota del alumno, entre 0 y 10: ", 0);
}
// A partir de aquí podemos procesar la nota ya correcta.
alert("La nota " + Nota + " es correcta");
</script>
```

Observaciones:

- Hay que asegurarse de que en algún momento se produzca la salida del bucle ya que de lo contrario estaríamos ante un bucle infinito. Por ejemplo, si en lugar de la condición `while (Nota < 0 || Nota > 10)` hubiésemos escrito: `while (Nota > 0 || Nota < 10)` el bucle hubiese estado iterando constantemente y no finalizaría nunca.
- Para que esta salida se produzca se deberá modificar el valor de la expresión. En nuestro caso la modificación se produce al leer un nuevo valor del teclado.

- Se puede forzar una salida del bucle en cualquier momento mediante la cláusula **break**.
- También se puede forzar a realizar un nuevo ciclo aún sin terminar todas las instrucciones del bucle mediante la cláusula **continue**.

Pero ninguna de estas dos últimas opciones es recomendable ya que dificultan la legibilidad de los programas y, por tanto, su posterior mantenimiento.

ACTIVIDAD PROPUESTA 8.11

- Utilizando el código anterior, añade las instrucciones necesarias para leer la variable sexo y validar que sea un valor correcto, H o h para Hombre y M o m para Mujer. Visualiza con `document.write` los valores correctos. Guarda el ejercicio con el nombre *Ejemplo13_1.html*.
- Crea una página que incluya un script para leer números de teclado y visualizarlos en la página. Leer números hasta que sea un 0, cuando el número leído sea un 0 finalizar el bucle. El bucle se ejecutará mientras el número leído sea distinto de 0. Guarda el ejercicio con el nombre *Ejemplo13_2.html*.

8.5.2 Estructuras *do...while*

La estructura *do...while* es similar a la anterior, pero en este caso la comprobación se produce después de ejecutar el bloque de instrucciones. El diagrama de flujo se representa como se muestra en la Figura 8.16 y el formato es el siguiente:

```
do
{
    Instrucciones;
}
while(condición);
```

La única diferencia entre la estructura *while* y la estructura *do...while* está en la primera vez que se ejecuta el bucle:

- la estructura **while** comprueba la condición antes de entrar por primera vez en el bucle y si la condición no se cumple, o no entrará.
- la estructura **do...while** ejecuta el bucle la primera vez sin comprobar la condición.

Para las demás iteraciones el funcionamiento es idéntico en ambas estructuras (únicamente se producen variaciones en el caso de utilizar la cláusula *continue*).

El siguiente ejemplo (*Ejemplo14.html*) utiliza un bucle *do...while* para resolver el ejercicio del *Ejemplo13.html*, pero ahora utilizando *do...while*.

```
<script type="text/javascript">
    // Ejemplo de estructura do...while
    var Nota;
    do
    {
        Nota = prompt("Introduce la nota del alumno:");
    , 0);
        if (Nota < 0 || Nota > 10)
```

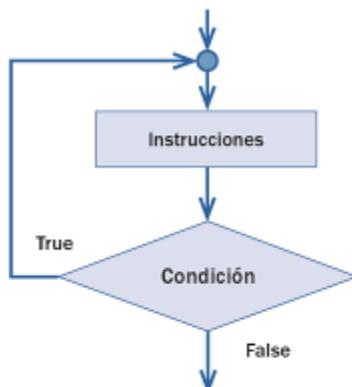


Figura 8.16. Estructura *do...while*.

**SABÍAS QUE:**

Código objeto. Es el código resultante de compilar el código fuente. No es directamente ejecutable por el ordenador ni entendido por el ser humano. Es un código o representación intermedia de bajo nivel.

Código ejecutable. Es el resultado de enlazar el código objeto con una serie de rutinas y bibliotecas, obteniendo así el código que es directamente ejecutable por la máquina.

```

{
    alert("Nota errónea " + Nota + ", teclea de nuevo. ");
}
while (Nota < 0 || Nota > 10);
alert("La nota " + Nota + " es correcta");
</script>

```

ACTIVIDAD PROPUESTA 8.12

- Utilizando el código del ejemplo anterior y la estructura do...while, añade las instrucciones necesarias para leer la variable sexo y validar que sea un valor correcto, H o h para Hombre y M o m para Mujer. Visualiza con document.write los valores correctos. Guarda el ejercicio con el nombre Ejemplo14_1.html.
- Crea una página que incluya un script para leer números de teclado y visualizarlos en la página. Leer números hasta que sea un 0, cuando el número leído sea un 0 finalizar el bucle. El bucle se ejecutará mientras el número leído sea distinto de 0. Haz el ejercicio utilizando un bucle do...while. Guarda el ejercicio con el nombre Ejemplo14_2.html.

Utilizando contadores y acumuladores

Un **acumulador** es una variable que se utiliza para ir acumulando valores. Por ejemplo, se declara la variable Suma para que sume los números leídos, la variable Suma es un acumulador, con la que se realizan las siguientes operaciones:

1. Inicialización: el acumulador debe ser inicializado (Suma = 0) antes de comenzar su función.
2. Acumulación: normalmente se realiza dentro de una estructura repetitiva y consiste en incrementar el acumulador con el valor de lo que se sume, por ejemplo números (Suma = Suma + Num).

Un **contador** es una variable que se utiliza para ir contando las veces que ocurre algo. Por ejemplo, se declara la variable Cuenta para que cuente los números que leemos, la variable Cuenta es un contador, con la que se realizan las siguientes operaciones:

1. Inicialización: el contador debe ser inicializado (Cuenta = 0) antes de comenzar su función.
2. Cuenta: normalmente se realiza dentro de una estructura repetitiva y consiste en incrementar el contador en 1 (Cuenta = Cuenta + 1).

El siguiente código (*Ejemplo15.html*) leerá una secuencia de números introducidos por teclado, hasta que se introduzca el 0, y calcula y visualiza la suma de esos números y el número de números leídos:

```

<script type="text/javascript">
    // Ejemplo de acumulador y contador
    var Num, Suma = 0, Cuenta=0;
    Num = parseInt(prompt("Introduce un número (cero para finalizar): ", 0));

```

```

while (Num != 0)
{
    Suma = Suma + Num;
    Cuenta = Cuenta + 1;
    Num = parseInt(prompt("Introduce un número
(cero para finalizar): ", 0));
}
document.write("<h3>La suma de los números es:" +
Suma + "</h3>");
document.write("<h3>El número de números leídos
es:" + Cuenta + "</h3>");
</script>

```

El siguiente código (*Ejemplo16.html*) es igual que el anterior pero devolverá, además, la media de los números tecleados. Para calcular la media se necesita contar el número de valores que han sido introducidos y sumarlos. Así, la media es el resultado de dividir la suma de los números leídos entre el contador de números leídos. Antes de dividir hay que comprobar que el contador de números sea mayor que 0, porque si es 0 dará error al dividir por 0. Las medias se calculan cuando finaliza el bucle.

```

<script type="text/javascript">
// Ejemplo de acumulador y contador
var Num, Suma = 0, Cuenta=0, Media;
Num = parseInt(prompt("Introduce un número (cero
para finalizar): ", 0));
while (Num != 0)
{
    Suma = Suma + Num;
    Cuenta = Cuenta + 1;
    Num = parseInt(prompt("Introduce un número
(cero para finalizar): ", 0));
}
document.write("<h3>La suma de los números es:" +
Suma + "</h3>");
document.write("<h3>El número de números leídos
es:" + Cuenta + "</h3>");
// Calculamos la media,
// pero antes preguntamos si se han leído números
// Si no se han leído números ponemos a 0 la media
if (Cuenta > 0)
{
    Media = Suma / Cuenta;
}
else
{
    Media = 0;
}
document.write("<h3>La media de los números es:" +
Media + "</h3>");
</script>

```



SABÍAS QUE:

El objeto **window** utilizado en JavaScript es el objeto básico, es decir el elemento principal del modelo de objetos, y representa en sí mismo al propio navegador y las ventanas que se abren. Los métodos principales de este objeto son: ***open()***, ***close()***, ***alert()***, ***confirm()***, ***prompt()***.

SABÍAS QUE:

La sintaxis básica para utilizar el método `window.open` es la siguiente:

```
window.open(URL, nombre)
```

En `URL` se especifica la dirección a abrir.

En `nombre` se puede especificar el nombre de la ventana o también los siguientes atributos de destino:

`_blank` – La URL se carga en una nueva ventana, es el valor por defecto.

`_parent` – La URL se carga dentro de la ventana padre.

`_self` – La URL sustituye a la página actual.

`_top` – La URL sustituye a todas las páginas que fueron cargadas.

ACTIVIDAD PROPUESTA 8.13

- Resuelve el ejercicio anterior utilizando un bucle `do...while`. Guarda el ejercicio como `Ejemplo16_1.html`.
- Crea una página que incluya un script que lea cinco números de teclado y que visualice la suma de ellos. Utiliza un bucle `do...while`. Guarda el ejercicio como `Ejemplo16_2.html`.
- Resuelve el ejercicio anterior utilizando un bucle `while`. Guarda el ejercicio como `Ejemplo16_3.html`.
- Crea una página que incluya un script para leer el nombre y la edad de tres alumnos. Y calcula la edad media de los alumnos. Al leer la edad hay que validarla comprobando que esté entre 13 y 100, utiliza para ello un bucle, `do...while`, o `while`. Guarda el ejercicio como `Ejemplo16_4.html`.
- Realiza los cambios necesarios al ejercicio anterior para que lea también el sexo del alumno, H o h para hombres y M o m para mujeres. Valida esta entrada. Guarda el ejercicio como `Ejemplo16_5.html`.
- Realiza los cambios necesarios al ejercicio anterior para que cuente el número de hombres y el número de mujeres que se han leído, y los visualice. Guarda el ejercicio como `Ejemplo16_6.html`.
- Realiza los cambios necesarios al ejercicio anterior para que lea también la nota del alumno. Valida esta entrada haciendo que la nota sea válida si está entre 0 y 10. Y que visualice también la nota media. Guarda el ejercicio como `Ejemplo16_7.html`.

8.5.3 Estructura for

La estructura `for` ejecuta un bucle un número determinado de veces controlando automáticamente el número de repeticiones o iteraciones. La utilizaremos siempre que sepamos previamente el número de veces que se ejecutará el bucle. Su formato genérico es el siguiente:

```
for (VariableControl = ValorInicio;
CondiciónContinuación; ExpresiónIncremento)
{
    Instrucciones;
    ...
}
```

Donde:

- `VariableControl`: es una variable interna o local al bucle (no hay que declararla previamente) que se utiliza normalmente como contador del número de ejecuciones del bucle en cada momento.
- `ValorInicio`: es el valor inicial que tomará la `VariableControl`.
- `CondiciónContinuación`: es una condición que será comprobada antes de realizar cada ejecución del bucle. Si se cumple, se ejecutará el bloque de instrucciones; en caso contrario, pasará el control a la línea siguiente al final de la estructura.
- `ExpresiónIncremento`: es una expresión que modificará el valor de la variable de control. Normalmente se trata de una simple suma pero puede ser cualquier expresión que permita en algún momento la salida del programa.

Veamos un ejemplo sencillo.

```
<script type="text/javascript">
    // Ejemplo de estructura for
    for (i = 1; i <= 3; i = i + 1)
    {
        document.write(i + "*");
    }
</script>
```

Este ejemplo (*Ejemplo17.html*) visualiza en la página el valor del índice del bucle, el bucle se ejecuta tres veces, desde *i* igual a 1 hasta *i* igual a 3, incluido. El índice se incrementa de uno en uno. Observa que:

- La variable de control es *i* y su valor de inicio es 1.
- La condición de continuación es que *i* sea menor o igual que 3.
- La expresión de incremento es *i* = *i* + 1.
- El bucle, por tanto, se ejecutará tres veces.

El resultado de la ejecución del programa será **1*2*3***.

La estructura *for* puede sustituirse por una estructura *while* siempre que tengamos en cuenta lo siguiente:

- Habrá que crear e inicializar antes de entrar en el bucle una variable que hará las funciones de variable de control.
- La condición del bucle *while* será la misma que la equivalente del bucle *for*.
- Al final del bloque de instrucciones habrá que incluir una instrucción adicional que será la encargada de incrementar (o variar de alguna manera) el valor de la variable que controla la ejecución del bucle.

Así, podemos escribir el programa anterior empleando una estructura *while*. El resultado será exactamente el mismo:

```
<script type="text/javascript">
    // Ejemplo de estructura while simulando un for
    var i = 1;
    while (i <= 3)
    {
        document.write(i + "*");
        i = i + 1;
    }
</script>
```

El siguiente código (*Ejemplo18.html*) lee un número de teclado y calcula y visualiza su factorial. Sabiendo que el factorial de un número N se calcula multiplicando los números desde 1 hasta él mismo incluido, es decir factorial de $N = 1 * 2 * 3 \dots * N$. Si el número N es 0, el factorial de N es 1. El factorial acumula productos con lo que es necesario iniciarlo a 1, si se inicializa a 0, al multiplicar por 0, nos devolverá 0.

IDEA:

Prueba el siguiente ejemplo de uso de los atributos de *window.open()*:

```
<!DOCTYPE html>
<html></head>
<body>
<p>Pulsa el botón abrir una página.</p>
<button onclick="window.open('http://www.google.es')">Ventana Nueva</button>
<button onclick="window.open('http://www.google.es','_parent')">Ventana Padre</button>
<button onclick="window.open('http://www.google.es','_self')">Ventana Self</button>
<button onclick="window.open('http://www.google.es','_top')">Ventana Top</button>
</body></html>
```

IDEA:

Prueba este ejemplo para ver la diferencia entre el uso de `alert()` y `confirm()`:

```
<script>
  var a=alert("Esto es una
  alerta");
  var c=confirm("Esto es otro
  tipo de alerta");
  document.write("<br/>Valor
  de alert: " + a);
  document.write("<br/>Valor
  de Confirm: " + c);
</script>
```

```
<script type="text/javascript">
  // Ejemplo factorial
  var Num = 0, Fact = 1;
  Num = parseInt(prompt("Introduce número:", 0));
  for (i = 1; i <= Num; i = i + 1)
  {
    Fact = Fact * i;
  }
  alert("Factorial de " + Num + " = " + Fact);
</script>
```

ACTIVIDAD PROPUESTA 8.14

- Crea una página que incluya un `script` que lea un número de teclado y visualice la tabla de multiplicar del 1 al 10 del número leído. Guarda el ejercicio como `Ejemplo18_1.html`. Utiliza un bucle `for` de 1 a 10 para ir visualizando las distintas multiplicaciones.
- Cambia el código anterior para que la tabla de multiplicar aparezca en la pantalla como se muestra en la Figura 8.17. Guarda el ejercicio como `Ejemplo18_2.html`.

NUMERO	FACTOR	RESULTADO
689	1	689
689	2	1378
689	3	2067
689	4	2756
689	5	3445
689	6	4134
689	7	4823
689	8	5512
689	9	6201
689	10	6890

Este está dentro del body.

Figura 8.17. Tabla de multiplicar del `Ejemplo18_2.html`.

- Cambia el código anterior para que las filas pares se muestren de un color y las impares de otro (véase Figura 8.18). Guarda el ejercicio como `Ejemplo18_3.html`. Para saber si una línea es par o impar basta obtener el resto de dividir el índice del bucle por 2, si el resto es 0 es par, si no es 0 es impar. La función resto es el carácter % (`resul = a%b`, devuelve el resto de dividir a por b).



SABÍAS QUE:

Los *cookies* son datos almacenados en pequeños ficheros de texto dentro del ordenador. Se inventaron para que los servidores recordasen la información del usuario que se conecta a un sitio web, así la siguiente vez que se conecte dispondrá de su información. Cuando un usuario se conecta a un sitio web el *cookie* almacena sus datos, la siguiente vez el *cookie* recordará esa información.

Con JavaScript se puede crear un *cookie*, por ejemplo la *cookie* *username* con ese valor:

```
document.cookie = "username=Alicia
Ramos";
```

Se puede leer un *cookie*:

```
var mycookie = document.cookie;
```

Se puede borrar un *cookie* añadiendo una fecha pasada de expiración:

```
document.cookie = "username=;
expires=Thu, 01 Jan 2010 00:00:00 GMT";
```

TABLA DE MULTIPLICAR DE 1224		
NUMERO	FACTOR	RESULTADO
1224	1	1224
1224	2	2448
1224	3	3672
1224	4	4896
1224	5	6120
1224	6	7344
1224	7	8568
1224	8	9792
1224	9	11016
1224	10	12240

Esto está dentro del body.

Figura 8.18. Tabla de multiplicar del Ejemplo18_3.html.

- Modifica el ejercicio Ejemplo16_7 de la Actividad propuesta 8.13. Utiliza un bucle for para leer los alumnos, en este ejercicio que se lean cinco alumnos y haz que la salida se muestre como se indica en la Figura 8.19. Guarda el ejercicio como Ejemplo18_4.html.

TABLA DE ALUMNOS			
NOMBRE	EDAD	SEXO	NOTA
Pedro García	18	h	6
Juan Antonio Chico	18	h	9
Alicia García	18	m	7
Ana Ramos	18	m	8
Pedro Matape	17	h	7

DATOS ESTADÍSTICOS.
La media de edad es : 17.8
La media de nota es : 7.4
Número de hombres: 3
Número de mujeres: 2

Esto está dentro del body.

Figura 8.19. Tabla de alumnos del Ejemplo18_4.html.

- Crea una página que incluya un script que lea dos números N1 y N2. Valida que los números sean positivos. Suma N1 el número de veces que indique N2. Visualiza la suma. Por ejemplo, si N1 es 5 y N2 es 3, hay que sumar N1 (el 5) tres veces (valor de N2), el resultado será 15 (5 + 5 + 5). Guarda el ejercicio como Ejemplo18_5.html.

SABÍAS QUE:



Con JavaScript se pueden crear objetos para almacenar fechas y horas utilizando el objeto Date(). Los formatos que se utilizan son los siguientes:

```
new Date() // fecha y hora actual
new Date(Cadena con la fecha)
new Date(año, mes, día, horas, minutos, seg., miliseg.)
```

Al mes enero le corresponde el número 0, a diciembre el 11.

Ejemplos de uso:

```
var hoy = new Date();
var dia1 = new Date("April 13, 2010 00:00:00");
var dia2 = new Date(2014,2,10);
var dia3 = new Date(2014,2,14,10,45,50);
document.write("<br />hoy: " + hoy);
document.write("<br />dia1: " + dia1);
document.write("<br />dia2: " + dia2);
document.write("<br />dia3: " + dia3);
```

8.6 Funciones útiles para manejar cadenas de texto

A medida que se va aprendiendo la programación en un lenguaje se hace necesario utilizar funciones predefinidas propias del lenguaje muy útiles para manejar números y caracteres. Algunas de las funciones más utilizadas para manejar cadenas son las siguientes:

- **length:** calcula la longitud de una cadena de texto, es decir el número de caracteres que la forman. Por ejemplo, definimos la variable *mensaje* con el valor "Hola mundo", y definimos la variable *lon*, en donde se almacenará la longitud:

```
var mensaje = "Hola mundo", lon;
lon = mensaje.length; // en lon se guarda 10
```

- **toUpperCase():** transforma los caracteres de la cadena a sus correspondientes caracteres en mayúsculas. Por ejemplo, definimos la variable *mensaje* con el valor "Hola mundo", y definimos la variable *mayusculas*, en donde se almacenará el mensaje en mayúsculas:

```
var mensaje = "Hola mundo", mayusculas;
mayusculas = mensaje.toUpperCase(); //mayusculas
= "HOLA MUNDO"
```

- **toLowerCase():** transforma los caracteres de la cadena a sus correspondientes caracteres en minúscula. Por ejemplo, definimos la variable *mensaje* con el valor "HOLA Mundo", y definimos la variable *minuscula*, en donde se almacenará el mensaje en minúsculas:

```
var mensaje = "HOLA Mundo", minusculas;
minusculas = mensaje.toLowerCase(); // minusculas
= "hola mundo"
```

- **charAt(posición):** obtiene el carácter que se encuentra en una cadena en la posición indicada en *posición*. La primera posición de una cadena es la posición 0, que apunta al primer carácter. Por ejemplo, definimos la variable *mensaje* con el valor "Hola mundo", y definimos la variable *letra* en donde se almacenará el carácter de *mensaje* indicado en la posición:

```
var mensaje = "Hola mundo", letra;
letra = mensaje.charAt(0); // letra = H, posición primera
letra = mensaje.charAt(3); // letra = a, posición cuarta
letra = mensaje.charAt(6); // letra = u, posición séptima
```

- **indexOf(carácter):** devuelve la posición en la que se encuentra el carácter indicado dentro de la cadena de texto. Si el carácter aparece

varias veces dentro de la cadena, se devuelve la primera posición empezando a buscar desde la izquierda. Si la cadena no contiene el carácter, la función devuelve el valor -1:

```
var mensaje = "Hola mundo", pos1;
pos1 = mensaje.indexOf('H'); // pos1 = 0
pos1 = mensaje.indexOf('m'); // pos1 = 5
pos1 = mensaje.indexOf('K'); // pos1 = -1, No existe
pos1 = mensaje.indexOf('o'); // pos1 = 1, La primera vez
```

- **lastIndexOf(carácter):** devuelve la última posición en la que se encuentra el carácter indicado dentro de la cadena de texto. Si la cadena no contiene el carácter, la función devuelve el valor -1:

```
var mensaje = "Hola mundo", pos1;
pos1 = mensaje.lastIndexOf('a'); // pos1 = 3
pos1 = mensaje.lastIndexOf ('o'); // pos1 = 9,
La última vez
```

El siguiente código (*Ejemplo19.html*) lee una cadena de caracteres de teclado y la visualiza en la página al revés, es decir del último al primer carácter.

```
<script type="text/javascript">
// Ejemplo de cadena al revés

var Cad1 = "";      // Cad1 es la cadena que se leerá.
var Cad2 = "";      // Cad2 contendrá la cadena al revés.
var Longitud = 0;   // Longitud guardará la longitud de la cadena.

Cad1 = prompt("Introduce una cadena:", " ");
Longitud = Cad1.length;

for (i = Longitud - 1; i >= 0; i = i - 1)
{
    Cad2 = Cad2 + Cad1.charAt(i);
}
document.write(Cad2);
</script>
```

Observamos que el programa, después de leer la cadena, va cogiendo uno a uno los caracteres de esta, comenzando por el final y terminando por el principio, y va construyendo de esta forma la nueva cadena (*Cad2*). La estructura *for* empleada en esta ocasión es decreciente, esto es, comienza con un valor igual a la longitud de la cadena menos uno (no olvidemos que mientras *length* devuelve la longitud total de la cadena, *charAt* empieza a contar desde el 0). De esta forma, si ejecutamos el programa e introducimos la cadena "HOLA" obtendremos el siguiente resultado: ALOH.

IDEA!

Para extraer de la fecha el día, el mes, el año, los minutos, los segundos, y las horas utilizaremos las siguientes funciones:

```
var hoy=new Date();
var h=hoy.getHours();
var m=hoy.getMinutes();
var s=hoy.getSeconds();
var any=hoy.getFullYear();
var mes=hoy.getMonth()+1;
var dia=hoy.getDay();
document.write("<br/>Año: " + any);
document.write("<br/>Mes: " + mes);
document.write("<br/>Dia: " + dia);
document.write("<br/>Hora: " + h);
document.write("<br/>Minutos: " + m);
document.write("<br/>Segundos: " + s);
```

IDEA:

Prueba el siguiente ejemplo, para abrir una ventana *popup* con la página de Google.

```
<!DOCTYPE html>
<html><body>
<button
  onclick="abrir('http://www.google.es')"
  >Abre ventana popup</button>

<script>
function abrir(url)
{
  link = window.open(url,"Link",
  "toolbar=0,location=0,directories=0,status=0,menubar=0,
  scrollbars=yes,resizable=0,
  width=430,height=300,left=80,
  top=180");
}
</script>
</body></html>
```

ACTIVIDAD PROPUESTA 8.15

- Crea una página que incluya un script que lea una cadena y visualice el número de vocales que contiene. Utiliza las funciones anteriores (*length* para saber la longitud de la cadena, y *charAt* para extraer un carácter y preguntar si es vocal *a, e, i, o, u*, en mayúscula y minúscula). Guarda el ejercicio como *Ejemplo19_1.html*.
- Crea una página que incluya un script que lea tres nombres de alumnos y los visualice en mayúscula, en minúscula, y que cuente las vocales (no acentuadas) de cada nombre. Visualiza los nombres en una tabla como se muestra en la Figura 8.20. Guarda el ejercicio como *Ejemplo19_2.html*.

NOMBRE ALUMNO	VOCALES	MAYÚSCULA	MINÚSCULA
Alicia San Juan	7	ALICIA SAN JUAN	alicia san juan
Alberto Carrasco	6	ALBERTO CARRASCO	alberto carrasco
Juan Antonio Marrupe	9	JUAN ANTONIO MARRUPE	juan antonio marrupe

Este está dentro del body.

Figura 8.20. Tabla de alumnos del *Ejemplo19_2.html*.

- Modifica el ejercicio anterior para que además visualice en la tabla los nombres de los alumnos al revés. Guarda el ejercicio como *Ejemplo19_3.html*.

8.7 Funciones en JavaScript

Una función es un conjunto de instrucciones a las que se asigna un nombre, devuelve o no un valor y se puede ejecutar tantas veces como se desea con solo llamarla con el nombre asignado. Se pueden definir tantas funciones como sea preciso.

Las funciones permitirán que se ejecuten una serie de instrucciones cuando se produce un evento (por ejemplo, cuando se pulsa un botón). También permitirán que se ejecute el mismo código sin necesidad de insertar la secuencia de instrucciones repetidas veces.

A la hora de trabajar con funciones debemos distinguir dos aspectos: la creación de la función (o definición) y la utilización de la función (o llamada). Para crear una función la tenemos que declarar dentro del código de nuestra página, y tendremos en cuenta:

- El nombre que vamos a poner a la función.
- Los parámetros o valores que va a recibir y sobre los que actuará, es decir, los argumentos.
- Las acciones que deberá realizar.
- El valor que devolverá. Aunque puede no devolver ningún valor.

El formato para declarar una función es el siguiente:

```
function nombredefuncion (listadeargumentos)
{
  instrucciones...;
```

```
...
[return valordretorno;]
}
```

Donde:

- *nombredefuncion* es un identificador válido que servirá para realizar llamadas a la función.
- *listadeargumentos* es una lista de variables (separadas por comas) que recogerán los valores que se pasen en la llamada a la función, si no hay argumentos no se pone nada.
- *valordretorno* es una expresión cuyo valor devolverá la función .

Las funciones pueden declararse en cualquier parte de una página HTML entre las etiquetas `<script>` y `</script>` pero teniendo en cuenta que no pueden definirse dentro de otra función ni dentro de una estructura de control. Normalmente, se definen dentro de la cabecera de una página HTML, así cuando se carga la página en el navegador del cliente estarán disponibles para ser utilizadas.

El siguiente ejemplo define una función llamada *suma* que recibe dos valores y devuelve su suma:

```
function suma(A,B)
{
    var C;
    C = A + B;
    return C;
}
```

Cuando el navegador encuentra la definición de una función, carga dicha función en la memoria pero no la ejecutará hasta que se produzca una llamada a la función.

Las llamadas a la función hacen que se ejecute el código de la función, deben hacerse siguiendo el formato:

```
nombredefuncion(listadeparámetros)
```

Hay que tener en cuenta que la llamada puede devolver un valor (si la función contiene la orden ***return***) y que debemos hacer algo con él, como escribirlo, asignarlo a una variable, etc. Por ejemplo, para hacer una llamada a la función ***suma*** y escribir el valor devuelto en el documento actual escribiremos:

```
document.write(suma(2,3));
```

En el siguiente código, *Ejemplo20.html*, se declaran dos funciones de suma,

- ***sumar1***: que recibe como argumento dos números (*a* y *b*), los suma y guarda la suma en *c*, y devuelve la suma al poner *return c*.



SABÍAS QUE:

El objeto ***document*** es el que tiene el contenido de toda la página que se está visualizando. Esto incluye el texto, las imágenes, los enlaces, los formularios, etc., gracias a este objeto podremos añadir dinámicamente contenido a la página, o hacer cambios, según nos convenga. Algunos de los métodos de ***document*** son:

clear(): borra el contenido del documento.
close(): cierra el *buffer* de escritura sobre el documento actual.

write(): escribe texto en el documento.
writeln(): idéntico a ***write()*** pero inserta un salto de línea al final.

getElementById(): selecciona un elemento según su atributo "id" especificado en el código HTML.

IDEA!

Prueba el siguiente ejemplo, dependiendo del botón pulsado en la alerta se visualiza un mensaje u otro. Se utiliza el método **innerHTML** para cambiar el contenido del elemento **id="prueba"**. Estudia su ejecución:

```
<!DOCTYPE html>
<html>
<body>
<p>Pulsa el botón para
visualizar la alerta.</p>
<button
onclick="funcion()">Prueba/<
button>
<p id="prueba"></p>
<script>
function funcion()
{ var mensaje;
  var r=confirm("Pulsa
el botón.");
  if (r==true) {
    mensaje="Has
pulsado OK.";
  }
  else {
    mensaje="Has pulsado
Cancel.";
  }
  document.
  getElementById("prueba").
  innerHTML=mensaje;
}
</script>
</body></html>
```

- **sumar2**: que recibe como argumento dos números (a y b), los suma y guarda la suma en c, pero en este caso no devuelve la suma, sino que escribe en el documento el resultado.

```
<!DOCTYPE html>
<head>
<title>Ejemplo 20</title>
<script type="text/javascript">
function sumar1(a,b)
{
  var c;
  c = a + b;
  return c;
}
function sumar2(a,b)
{
  var c;
  c = a + b;
  document.write("<h3>La suma de " + a + " + " + b
+ " = " + c +"</h3>");
}
</script>
</head>
<body>
<p>CUERPO DE LA PÁGINA</p>
<script>
  document.write("Llamada a la función con
argumentos (2+3) = ");
  document.write(sumar1(2,3));
</script>
<p onclick='sumar2(20,30);'> Haz clic en este
párrafo para sumar 20 + 30.</p>
</body></html>
```

Observa cómo se llama en cada caso a las funciones. En el primer caso se hace dentro de un script y para llamar a **sumar1** lo hacemos dentro de **document.write** para que visualice lo que devuelve la función. También se puede guardar el resultado de la llamada en una variable y luego visualizar la variable:

```
<script>
  var resul;
  document.write("Llamada a la función con
argumentos (2+3) = ");
  resul=sumar1(2,3);
  document.write(resul);
</script>
```

En el segundo caso la llamada a **sumar2** lo hacemos dentro de un evento **on-click** que lo que hace es llamar a la función y ejecutarla, en este caso no devuelve nada, la función solamente se ejecuta y en la ejecución se visualiza la suma.

Si al llamar a una función se le pasa un número menor de argumentos, los que no han obtenido valor como resultado de la llamada quedarán con el valor `null`.

Por ejemplo, si la llamada a la función `sumar` se realiza con un solo número el resultado será: `NaN`. Para evitar este tipo de situaciones podemos comprobar el número de argumentos que se ha pasado a una función mediante `arguments.length`, la función con la comprobación quedará así:

```
function sumar(a,b)
{
    var c;
    if (arguments.length != 2)
    {
        alert(" Mal el número de argumentos.");
        a=0;b=0;
    }
    c = a + b;
    return c;
}
```

En el siguiente ejemplo, `Ejemplo21.html`, se declara la función `dibujatabla()`, y dentro del `body` añadimos un botón, para que al pulsarlo se visualice en la pantalla la tabla.

```
<!DOCTYPE html>
<head>
<title>Ejemplo 21</title>
<script type="text/javascript">
function dibujatabla()
{
    document.write("<h2>Dibujo de una tabla de 3x3
</h2>");
    document.write("<table border=1>");
    document.write('<tr id="fila1"><th>Título 1</
th><td>Celda 1</td><td>Celda 2</td></tr>');
    document.write('<tr id="fila2"><th>Título 2</
th><td>Celda 3</td><td>Celda 4</td></tr>');
    document.write('<tr id="fila3"><th>título 3</
th><td>celda 5</td><td>Celda 6</td></tr>');
    document.write('</table>');
}
</script>
</head>
<body>
<p>Esto está dentro del body.</p>
<p>Haz clic en el botón para dibujar la tabla.</p>
<form >
    <input type='button' value=' Dibuja Tabla. '
    onclick='dibujatabla();'>
</form>
</body></html>
```

SABÍAS QUE: ?

El **Document Object Model** o DOM (Modelo de objetos del documento o Modelo en objetos para la representación de documentos) es una interfaz de programación de aplicaciones (API) que proporciona un conjunto estándar de objetos para acceder, añadir y cambiar dinámicamente el contenido estructurado en documentos HTML y XML.

A través del DOM, los programas pueden acceder y modificar el contenido, la estructura y el estilo de los documentos HTML y XML, que es para lo que se diseñó principalmente.

http://es.wikipedia.org/wiki/Document_Object_Model

SABÍAS QUE:



Para acceder a un elemento de un formulario se puede hacer de muchas formas diferentes.

Por ejemplo, imaginemos que queremos acceder a un campo identificado con id= "**Nombre**" que forma parte del formulario identificado también con id= "**datos**", y que es el primer formulario de la página. Podemos hacerlo de varias formas. El formulario es este:

```
<form method="post" action="" name="datos" id="datos">
    Nombre: <input type="text" name="Nombre" id="Nombre" >
</form>
```

Para acceder al Nombre podemos escribir:

```
document.
getElementById("Nombre").
value;
document.forms.datos.
elements.Nombre.value;
document.forms[0].
Nombre.value;
document.datos.Nombre.value;
```

ACTIVIDAD PROPUESTA 8.16

- Modifica el código del ejercicio anterior para que al pulsar el botón visualice una tabla con seis enlaces. Véase Figura 8.21. Guarda el ejercicio como Ejemplo21_1.html.

Tabla de mis enlaces favoritos

Buscadores	Noticias	Música
Google	Diario El país	Musica los 40.com
Página de Yahoo	Diario El mundo	Musica Cadena 100.
Página de Altavista	Diario La tribuna de CLM	Musica M80 Radio.

Figura 8.21. Tabla del Ejemplo21_1.html.

- Modifica el código del ejercicio anterior para que al pulsar el botón visualice una tabla con cuatro imágenes de tu ciudad (búscalas en internet). Véase Figura 8.22. Guarda el ejercicio como Ejemplo21_2.html.

Tabla con mis fotos



Figura 8.22. Tabla del Ejemplo21_2.html.

- A partir de los ejercicios anteriores crea una nueva página Ejemplo21_3.html en la que aparezca un formulario con tres botones, y dependiendo del botón a pulsar que se dibujen las tablas realizadas en el Ejemplo21, Ejemplo21_1 y Ejemplo21_2. El formulario se muestra en la Figura 8.23. Añade después del dibujo de cada tabla un enlace para volver al ejercicio Ejemplo21_3.

Haz clic en el botón correspondiente.

Figura 8.23. Formulario de botones del Ejemplo21_3.html.

8.7.1 Acceso desde JavaScript a los elementos de un formulario

Cuando utilizamos formularios y se desea acceder a los elementos del formulario para comprobar los valores o para realizar una serie de acciones, disponemos de una serie de propiedades que nos van a permitir trabajar con el contenido tecleado en los elementos de un formulario.

El siguiente ejemplo (*Ejemplo22.html*) muestra cómo recoger los datos del formulario (véase Figura 8.24) y comprueba si los datos tecleados son correctos o no.

```
<!DOCTYPE html>
<head>
<title>Ejemplo 22</title>
<script type="text/javascript">
function comprobar()
{
    var nombre= document.miFormulario.nombre.value;
    var edad=document.miFormulario.edad.value;
    var grupo=document.miFormulario.grupo.value;
    document.write("<h2>Datos leídos en el
formulario.</h2>");
    document.write("<h3>Nombre: " + nombre);
    document.write("<br/>Edad: "+edad);
    document.write("<br/>Grupo: "+grupo+ "</h3>");
    //comprueba que se teclee algo en nombre
    if (nombre.length==0) alert("Tiene que escribir su
nombre.");
    //Comprueba la edad, tiene que ser entero mayor
    que 18
    if (edad=="")
        alert("Tiene que introducir una edad >= a
18.");
    else
        if (edad<18)
            alert("Debe ser mayor de 18 años.");
    //vuelve al formulario
    document.write("<h3> <a href='Ejemplo22.
html'>Volver atrás</a> </h3>");
}
</script>
</head>
<body >
<h2 align="center">FORMULARIOS Y JAVASCRIPT</h2>
<table border="1" align="center"
bgcolor="yellow"><tr><td>
<form name="miFormulario">
    <p>&nbsp;&nbsp;&nbsp;Nombre: <input type="text"
name="nombre" size="30"
        maxlength="100" id="nombre"
    />&nbsp;&nbsp;&nbsp;</p>
    <p>&nbsp;&nbsp;&nbsp;Edad: <input type="text"
name="edad" size="3"
        maxlength="2" /></p>
    <p> &nbsp;&nbsp;&nbsp;GRUPO: <select
name="grupo">
        <option value="1SMR">Primero SMR </option>
        <option value="2SMR">Segundo SMR </option>
        <option value="1DAM">Primero DAM </option>
        <option value="2DAM">Segundo DAM </option>
    </select></p>
```

8. LENGUAJE JAVASCRIPT



SABÍAS QUE:

Los eventos **DOM HTML** permiten registrar diferentes controladores de eventos en elementos de un documento HTML. Los eventos se utilizan normalmente en combinación con las funciones JavaScript, y la función no se ejecutará antes de que ocurra el evento, algunos de ellos son: **onclick, ondblclick, onmouseover, onkeypress, onload**. Ejemplos:

```
<body onload="funcion1()">
<button onclick="
funcion2()">Púlsame</button>
<input type="text"
onkeypress="funcion3()">
```

Podemos cambiar el estilo al pasar con el ratón:

```
<h1 onmouseover="style.
color='red'"
onmouseout="style.
color='blue'">Pasa
el ratón</h1>
```

Podemos cambiar el contenido con **this.innnerHTML** (**this** indica el elemento actual) y poner estilo al pasar con el ratón:

```
<h2 onmouseover="this.
innnerHTML='Paso';
style.color='red'"'
onmouseout="this.
innnerHTML='Salgo';
style.color='yellow'">
Pasa el ratón</h2>
```

FORMULARIOS Y JAVASCRIPT

Nombre:

Edad:

GRUPO: Primero SMR

Figura 8.24. Formulario del Ejemplo22.html.

```
<p align="center"><input type="button" value="Comprobar datos " onclick="comprobar();"/></p>
<hr/>&nbsp;&nbsp;&nbsp;<a href="javascript:comprobar();">Comprobar.</a>
</form></td></tr></table>
</body></html>
```

Para hacer referencia a los campos del formulario se utiliza la instrucción: `document.nombre_del_formulario.nombre_del_elemento.value`, donde

- **document** se refiere a la página que contiene el formulario.
- **nombre_del_formulario** es el nombre del formulario donde se encuentran los elementos, en el ejemplo es `miFormulario`.
- **nombre_del_elemento** es el nombre de cada uno de los controles del formulario, es decir es lo que ponemos en el atributo `name` de los controles. En el ejemplo son `nombre`, `edad` y `grupo`.
- **value** es la propiedad que devuelve el valor tecleado o seleccionado de los elementos del formulario.

Al pulsar el botón `Comprobar datos`, o al enlace `Comprobar`, se hace una llamada a la función `comprobar()` que lo que hace es recoger los datos del formulario en variables. Para llamar a una función JavaScript con un enlace lo pondremos así: `Enlace.`.

La función recoge los datos, los visualiza y comprueba que se teclee algo en el campo `nombre` y en el campo `edad`, y además comprueba que la edad sea mayor de 18. Si estas condiciones no se cumplen visualiza una alerta con el mensaje de error.

El siguiente ejemplo (*Ejemplo23.html*) realiza operaciones de cálculo a partir de dos números tecleados en un formulario. Las operaciones son la suma y la resta de dos números. Al pulsar el botón `Sumar los números` se ejecuta la función `sumar()`, y al pulsar el botón `Restar los números` se ejecuta la función `restar()`. El formulario se muestra en la Figura 8.25

OPERACIONES CON NÚMEROS

Introduce número1:

Introduce número2:

Figura 8.25. Formulario del Ejemplo23.html.

```
<!DOCTYPE html>
<head>
<title>Ejemplo 23</title>
<script type="text/javascript">
function sumar(){
    var num1 = document.miFormulario.num1.value;
    var num2 = document.miFormulario.num2.value;
    var suma;
    //
    if (isNaN(num1) || num1=="")
        { document.write(" <br/>El número 1 es incorrecto,
no es número: "+num1);
        num1=0; }
    if (isNaN(num2) || num2=="")
        { document.write(" <br/>El número 2 es incorrecto,
no es número: "+num2);
        num2=0; }
```

```

suma = parseInt(num1) + parseInt(num2);
document.write("<br/>num1 = "+num1+"<br/>num2 =
"+num2+"<br/>Suma = "+suma);
document.write("<h3> <a href='Ejemplo23.
html'>Volver atrás</a> </h3>");
}
function restar(){
var num1 = document.miFormulario.num1.value;
var num2 = document.miFormulario.num2.value;
var resta;
if (isNaN(num1) || num1=="")
{ document.write("<br/>El número 1 es incorrecto,
no es número: "+num1);
num1=0; }
if (isNaN(num2) || num2=="")
{ document.write("<br/>El número 2 es incorrecto,
no es número: "+num2);
num2=0; }
resta = parseInt(num1) - parseInt(num2);
document.write("<br/>num1 = "+num1+"<br/>num2 =
"+num2+"<br/>Resta = "+resta);
document.write("<h3><a href='Ejemplo23.
html'>Volver atrás</a></h3>");
}
</script>
</head>
<body>
<h2 align="center">OPERACIONES CON NÚMEROS</h2>
<table border="1" align="center"
bgcolor="cyan"><tr><td>
<form name="miFormulario">
<p>&ampnbsp&ampnbsp&ampnbspIntroduce número1:
<input type="text" name="num1" size="15"/></p>
<p>&ampnbsp&ampnbsp&ampnbspIntroduce número2:
<input type="text" name="num2" size="15"/></p>
<hr/><p align="center">
<input type="button" value=" Sumar los números "
onclick="sumar();"/>
<input type="button" value=" Restar los números "
onclick="restar();"/>
</p>
</form></td></tr></table>
</body></html>

```

En el ejemplo se utiliza la función ***isNaN***; esta función de JavaScript evalúa un argumento para determinar si este no es un número. La sintaxis para ***isNaN*** es: ***isNaN(Argumento_o_Variable)***. Donde ***Argumento_o_Variable*** es el valor que se quiere evaluar.

Las funciones ***parseFloat*** y ***parseInt*** retornan '***NaN***' cuando evalúan un valor que no es un número. Al poner esta sentencia ***If (isNaN(num2) || num2=="")*** se comprueba si el número tecleado (en este caso ***num2***) no es número o si es vacío. Si se cumple, en el ejercicio se visualiza un mensaje indicando que es incorrecto y se pone a 0.

ACTIVIDAD PROPUESTA 8.17

- Modifica el código del ejercicio anterior y añade dos botones más al formulario, uno para dividir los números tecleados y otro para multiplicarlos. Realiza una función para dividir y otra para multiplicar. Comprueba que los datos tecleados sean números y que no estén vacíos. En el caso de dividir (***num1 / num2***) comprueba antes que ***num2*** sea distinto de 0, si es igual a 0, no se puede dividir, en ese caso visualiza el mensaje. Guarda el ejercicio como ***Ejemplo23_1.html***.

- JavaScript es un lenguaje de script que se utiliza para añadir contenido dinámico a las páginas web. El código JavaScript va incrustado en las páginas HTML y los navegadores son los encargados de interpretar este código. El código JavaScript se puede incrustar de tres formas dentro del código HTML:
 - Incluir el script dentro de la etiqueta <head>.
 - Incluir el script en un archivo externo.
 - Incluir el script dentro del <body> en los elementos HTML o XHTML.
- El formato que utilizaremos para incrustar código JavaScript es:

```
<script type="text/javascript">
<!--
// aquí irá el código
//-->
</script>
```

- A diferencia de HTML y CSS, JavaScript sí diferencia entre mayúsculas y minúsculas. Las instrucciones JavaScript terminan con ; . Para añadir comentarios de una línea se utiliza //, y de varias líneas /* */.

Instrucciones básicas:

- Para leer los datos de teclado se utiliza la orden **prompt**: **prompt("mensaje","texto_por_defecto")**.
- Para visualizar en la página se utiliza **document.write("Mensaje a visualizar")**.
- Para poder visualizar una alerta se utiliza: **alert("Mensaje de la alerta")**.

Elementos básicos de programación

Variables	Elementos utilizados para almacenar la información. Ejemplos son: nombre, direcc, edad, tif. Las variables no deben llevar acentos. Recuerda que JavaScript es sensible a mayúsculas y minúsculas, con lo cual la variable Nombre es distinta de la variable nombre y de la variable NOMBRE.
Operadores	Símbolos que se utilizan para operaciones de cálculo y para comparaciones de valores. Los dividimos en tres grupos: <ul style="list-style-type: none"> • Aritméticos para calcular: suma (+), resta (-), multiplicación (*) y división (/). • Relacionales para comparar: mayor que (>), menor que (<), mayor o igual (>=), menor o igual (<=), igual que (==) y distinto de (!=). • Lógicos para operaciones complejas, devuelven True o False: o (OR), y (AND), y no (NOT).
Expresiones	Son combinaciones de variables, constantes y operadores que devuelven un valor realizando determinadas operaciones. Ejemplo: Suma = Suma +1000 If (NumeA > NumeB) Resta = Numa - Numb;
Sentencias o instrucciones	Una sentencia puede incluir cualquier elemento de la gramática de JavaScript.
Funciones	Secuencia de sentencias que tienen un nombre y que puede ejecutarse las veces que sea necesario llamándola por su nombre.
Acumuladores	Son variables que se utilizan para ir acumulando valores. El acumulador debe ser inicializado (Suma = 0) antes de comenzar su función, la acumulación se realiza dentro de una estructura repetitiva y consiste en incrementar el acumulador con el valor de lo que se sume; por ejemplo, números (Suma = Suma + Num).
Contadores	Son variables que se utilizan para ir contando las veces que ocurre algo. El contador debe ser inicializado (Cuenta = 0) antes de comenzar su función, y se va incrementando en 1 dentro de una estructura repetitiva (Cuenta = Cuenta + 1).

- Estructuras de programación: estructuras de control

```
Estructura if
if (condición)
{ // si se cumple la
  condición
  Instrucciones1;
  Instrucciones2;
}

Estructura if...else
if (condición)
{ // si se cumple la
  condición
  Instrucciones1;
  Instrucciones2;
  Instrucciones3;
}
else
{ // si no se
  cumple la condición
  Instrucciones4;
  Instrucciones5;
}
```

```
Estructura
alternativa múltiple
switch
  (expresión)
  { case valor1:
    instrucciones1;
    break;
    case valor2:
    instrucciones2;
    break;
    case valor3:
    instrucciones3;
    break;
    ...
    case valorn:
    instruccionesn;
    break;
    [default:
    instrucciones;]
```

- Funciones de cadena:

length: calcula la longitud de una cadena de texto, es decir el número de caracteres que la forman.

toUpperCase(): transforma los caracteres de la cadena a sus correspondientes caracteres en mayúsculas.

toLowerCase(): transforma los caracteres de la cadena a sus correspondientes caracteres en minúsculas.

charAt(posición): obtiene el carácter que se encuentra en una cadena en la posición indicada en *posición*.

indexOf(caracter): devuelve la posición en la que se encuentra el carácter indicado dentro de la cadena de texto.

lastIndexOf(caracter): devuelve la última posición en la que se encuentra el carácter indicado dentro de la cadena de texto.

- Una **función** es un conjunto de instrucciones a las que se asigna un nombre, devuelve o no un valor y se puede ejecutar tantas veces como se desee con solo llamarla con el nombre asignado. El formato para declarar una función es el siguiente:

```
function nombredefuncion
(listadeargumentos)
{
  instrucciones...
  ...
  [return valordeleretorno;]
```

- Estructuras de programación: estructuras repetitivas

```
Estructura while
while(condición)
{
  Instrucciones;
}
```

```
Estructura do...while
do
{
  Instrucciones;
}
while(condición);
```

```
Estructura for
for (VariableControl = ValorInicio;
CondiciónContinuación;
ExpresiónIncremento)
{
  Instrucciones;
  ...
}
```

- Para hacer referencia a los campos del formulario se utiliza la instrucción: **document.nombre_del_formulario.nombre_del_elemento.value**.

- Para llamar a una función JavaScript con un enlace se escribe de la siguiente manera: **Enlace.**.

- Para llamar a una función JavaScript desde un botón de un formulario se utiliza el evento **onclick**: **<input type="button" value="Texto del botón " onclick="función_a_ejecutar();"/>**.

ACTIVIDADES DE ENSEÑANZA Y APRENDIZAJE

DE COMPROBACIÓN

8.1 El lenguaje JavaScript:

- a) Añade contenido dinámico a las páginas web.
- b) Se utiliza para acceder a bases de datos.
- c) Sirve para dinamizar los gestores de contenidos y los blogs.

8.2 Dentro de qué elemento HTML ponemos el código JavaScript:

- a) <scripting>.
- b) <jss>.
- c) <javascript>.
- d) <script>.

8.3 ¿Cuál es la sintaxis correcta para referenciar a un script almacenado en un fichero externo llamado "funciones.js"?

- a) <script src=" funciones.js">.
- b) <script name=" funciones.js">.
- c) <script href=" funciones.js">.

8.4 Existen varias formas de integrar lenguaje JavaScript en las páginas:

- a) Dentro de la etiqueta body.
- b) Dentro de la etiqueta head.
- c) En un fichero aparte.
- d) Las tres anteriores son correctas.

8.5 En un programa JavaScript:

- a) Las variables se utilizan para ejecutarse las veces que sean necesarias.
- b) Las funciones almacenan los datos con los que se trabaja.

8.6 Los operadores se utilizan para calcular y comparar.

8.7 ¿Cuál es la declaración correcta?

- a) var nomb1, Nomb1, NOMB1;
- b) var prompt('Teclea el nombre:', nombre);
- c) var 'pepe', nombre, edad;

8.8 ¿Cómo se declara la función de nombre Calculadora?

- a) function=Calculadora()
- b) function Calculadora ()
- c) function: Calculadora ()

8.9 ¿Cuál es la orden correcta?

- a) If (variable) {variable=20; else variable=30;}
- b) If (variable) {variable=20;} else {variable=30;}
- c) If (variable) then {variable=20;} else {variable=30;}

8.10 Dados los valores m=10 y a=0, ¿cuántas veces se ejecuta este bucle while(m!=10) { a=a+2;};

- a) Ninguna vez.
- b) 10 veces.
- c) Infinitas veces.

8.10 Dados los valores A=10 y B=0 y después de la instrucción if (B!=A) {A=A+3;B=B+2;} else (A=3);, ¿cuáles serán sus valores?

- a) A=3 y B=0.
- b) A=13 y B=2.
- c) A=10 y B=0.

DE APLICACIÓN

8.11 Dado este código, ¿qué visualiza al ejecutarse?

```
var A = 10, B = 12, C = 5;
document.write("A==B es: " + (A==B));
document.write("A<=B && C==9: " + (A<=B && C==9));
document.write("A==C || C>=B: " + (A==C || C>=B));
```

8.12 Dado este código, ¿qué visualiza al ejecutarse?

```
var A = 10, B = 12, C = 0;
if (A==B)
    {C=C*10;B=B+C;}
else
    {C=30;A=B+C;}
document.write("<br />A= " + A + "<br />B= " + B + "<br />C= " + C );
```

8.13 Dado este código, ¿qué visualiza al ejecutarse?

```
var A=-100, resul=0;
if(A!=0)
{ resul= resul *A;
  if(resul >0) {A=200;resul=resul-A;}
  else {A=A+100;resul=resul-1;}
}
else
{A = A*A; resul = resul* A;}
A=A+1;
resul=resul+2;
document.write("<br />A= " + A + "<br />Resul= " + resul);
```

8.14 Dado este código, ¿cuántas veces se ejecutará el bucle? ¿Cuándo sale del bucle? ¿Qué visualiza al ejecutarse?

```
var tipo;
tipo=prompt("Teclea el tipo (A, B ,o C):",0);
while (tipo != "A" || tipo != "B" || tipo != "C")
{ tipo=prompt("Teclea el tipo :",0); }
document.write("<br /> El tipo es = " + tipo);
```

8.15 Dado este código, ¿cuántas veces se ejecutará el bucle? ¿Qué visualiza?

```
var numerol = 1;
var resultado = 0;
do
    { resultado=resultado - numerol; numerol=numerol + 1;}
while (numerol<=3);
document.write("<br /> Resultado es = " + resultado);
```

8.16 Dado este código, ¿qué se visualizaría si *numero* fuese igual a 3? ¿Y si fuese igual a 2?

```
var numero, resul=0;
var Calidad = " ";
switch(numero)
    { case 1: Calidad= "Extra"; resul=N1 + 10;
    case 2: Calidad = "Super"; resul= resul + 15;
        break;
    case 3:
    case 4: Calidad = "Ligera"; resul= resul + 20;
    case 5:
    case 6: Calidad = "Buena"; resul= resul + 25;
    default: Calidad = "Normal"; resul= resul + 10;
    };
document.write("<br />Valor de resul: " + resul);
document.write("<br />Valor de Calidad: " + Calidad);
```

Enlaces web de interés

Tutorial de JavaScript: <http://librosweb.es/javascript/>

Tutorial de JavaScript de la w3schools: <http://www.w3schools.com/js/default.asp>

Tutorial de JavaScript: <http://www.javascriptya.com.ar/>

Resumen de JavaScript: <http://www.javascript.su/>

Curso de JavaScript: <http://aprende-web.net/javascript/>

Aplicaciones web

Este libro desarrolla los contenidos del módulo profesional de Aplicaciones Web, del Ciclo Formativo de Sistemas Microinformáticos y Redes, de la familia profesional de Informática y Comunicaciones, según lo establecido por el Real Decreto 1691/2007, de 14 de diciembre.

La obra se organiza en 8 Unidades didácticas que cubren los contenidos que se indican a continuación. Las Unidades 1 y 2 permiten iniciarse en el conocimiento y el uso de internet, el lenguaje HTML y las hojas de estilo, con las que poder realizar páginas web. En las Unidades 3 y 4 se explica cómo instalar y administrar tanto sistemas gestores de contenidos como sistemas gestores de aprendizaje a distancia; también se aprende a crear sitios web basados en plataformas de gestión de contenidos y sitios web basados en gestores de aprendizaje a distancia en los que poder crear y gestionar cursos. A continuación, en las Unidades 5, 6 y 7 se introduce al alumno en el uso y la configuración de aplicaciones web o servicios disponibles en internet más conocidos como *Cloud Computing*; también se enseña a manejar escritorios online, aplicaciones ofimáticas web, de almacenamiento y todo tipo de aplicaciones online. Finalmente, en la Unidad 8 se realiza una introducción a la programación a través del lenguaje JavaScript. Esto permite al alumno iniciarse en el mundo de la programación de ordenadores, que le servirá como base para proseguir su formación profesional y acceder a los Ciclos Formativos de grado superior de Desarrollo de Aplicaciones Web y de Desarrollo de Aplicaciones Multiplataforma de la familia profesional de Informática y Comunicaciones.

Cada unidad presenta una pequeña introducción, una lista de contenidos y una de objetivos principales. A continuación, el desarrollo de los contenidos, que se presenta desde un punto de vista esencialmente práctico, se va intercalando con actividades propuestas y resueltas y recuadros de información adicional. Una vez concluida la explicación teórica, se incluye un útil resumen que permite al alumno repasar las nociones principales estudiadas antes de evaluar su aprendizaje y poner en práctica y ampliar sus conocimientos a través de las Actividades de enseñanza y aprendizaje. Por último, al final de cada unidad se ofrecen listados de enlaces web a los que acceder para profundizar la información.

Asimismo, el libro ofrece un conjunto de recursos digitales, a los que se puede acceder a través de la ficha web de la obra (en www.paraninfo.es) y mediante un sencillo registro desde la sección de "Recursos previo registro". Asimismo, como recursos para el profesor, el libro incluye la guía didáctica y el solucionario de las actividades; a todos ellos se puede acceder también desde la ficha web de la obra.

Las autoras, actualmente profesoras de Formación Profesional de Ciclos Formativos de la familia profesional de Informática y Comunicaciones, cuentan con una extensa experiencia profesional en este ámbito. Son diplomadas en Informática por la Universidad de Extremadura. Alicia Ramos Martín ha colaborado en la elaboración de las Cualificaciones Profesionales para la familia profesional de Informática y Comunicaciones. Por su parte, M.^a Jesús Ramos Martín es catedrática de Enseñanza Secundaria e ingeniera en Informática por la Universidad Nacional de Educación a Distancia. Además, ambas son autoras de varios libros de Ciclos Formativos de grado medio y de grado superior de la familia profesional de Informática y Comunicaciones.