aula MENTOR

GOBIERNO DE ESPAÑA

MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE

Iniciación a Javascript

AULA MENTOR

educacion.es

ISBN: 978-84-369-5433-3
Nipo: 030-12-327-8

Autores:
Jorge Mohedano
José Miguel Saiz
Pedro Salazar Román
Equipo de contenidos de Aula Mentor

Coordinación pedagógica:
Aula Mentor

Ilustración de portada:
María Guija Medina

Tabla de contenido

Tabla de Ejemplos

UNIDAD 1 - INTRODUCCIÓN

1.1 - ¿QUÉ ES JAVASCRIPT?

Javascript es un lenguaje de programación que se utiliza principalmente para crear páginas Web capaces de interactuar con el usuario. Las páginas Web se consideran estáticas cuando se limitan a mostrar un contenido establecido por su creador sin proporcionar más opciones al usuario que elegir entre los enlaces disponibles para seguir navegando. Cuando un creador incorpora JavaScript a su página, proporciona al usuario cierta capacidad de interactuación con la página Web, es decir, cierto dinamismo y por lo tanto se incrementan las prestaciones de la misma al añadir procesos en respuesta a las acciones del usuario. Es importante señalar que estos procesos se ejecutan en la máquina del cliente (en el navegador) y por tanto no implican intercambio de datos con el servidor. Con Javascript se accede al mundo de las páginas Web dinámicas.

Para profundizar en el concepto de página dinámica se recomienda la lectura del artículo siguiente: *http://www.desarrolloweb.com/manuales/7/*

Desde el punto de vista técnico Javascript es un lenguaje interpretado, eso significa que las instrucciones son analizadas en secuencia por el intérprete de Javascript del navegador Web, de manera que su ejecución es inmediata a la interpretación. Esto permite que, una vez escrito un programa en Javascript con un editor de texto plano y embebido el código en un fichero HTML, el navegador es capaz de interpretarlo y ejecutarlo sin necesidad de procesos intermedios. Es importante destacar que Javascript y Java son dos lenguajes distintos ya que la semejanza de los nombres a veces puede llevar a confusión.

Javascript	Java
Es un lenguaje sencillo.	Es un lenguaje más complejo y completo con una *estricta* orientación a objetos.
Diseñado para desarrollar aplicaciones Web.	Diseñado para desarrollar aplicaciones de propósito general incluyendo tanto aplicaciones de escritorio como Web.
Lenguaje interpretado por el navegador. No requiere compilador.	Lenguaje compilado a código intermedio que luego es interpretado.
Más flexible. Por ejemplo, no requiere declarar variables ni tipos (aunque es aconsejable hacerlo).	Tiene reglas mucho más rígidas. Por ejemplo, hay que declarar todas las variables con sus tipos.

Tabla 1 - Javascript vs Java

Para conocer más datos sobre la historia y la evolución de Javascript puede consultar *http://www.librosweb.es/javascript/capitulo1/breve_historia.html*

1.2 - EDITAR JAVASCRIPT

La edición de código escrito en lenguaje JavaScript no requiere de ninguna herramienta especial, siendo suficiente un editor de texto plano (sin formato). El código JavaScript debe integrarse en una página Web que será la que establezca el contexto de ejecución del código. Dicha página puede ser abierta con un navegador Web y como consecuencia de ello se ejecutará el programa JavaScript en base al propio flujo de control de la página Web. Esto significa que si el código JavaScript se asocia a la pulsación de un botón, dicho código se interpretará y ejecutará cuando el usuario pulse el botón. Esto implica que, como la interpretación y la ejecución son simultáneas, los errores en el código (salvo aquellos que imposibiliten la interpretación del código HTML) serán detectados en el momento de la ejecución. Esto complica en parte la depuración de los programas.

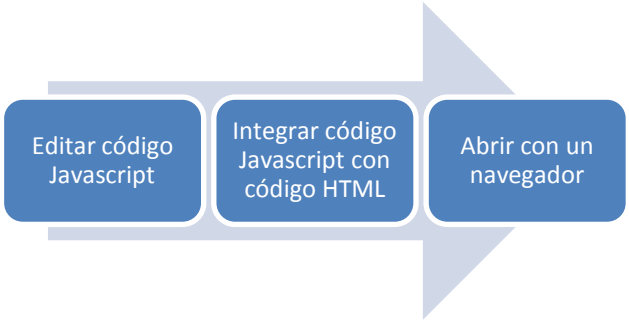


Ilustración 1 – Proceso de edición y ejecución de un programa Javascript

Para editar código Javascript y/o HTML existen multitud de editores. En el ámbito de los sistemas Windows uno de los más versátiles, potentes y gratuitos es NotePad++ *http://notepad-plus-plus.org/*

1.3 - MECANISMOS PARA INTEGRAR EL CÓDIGO JAVASCRIPT

1.3.1 - INCLUIR JAVASCRIPT DIRECTAMENTE EN EL DOCUMENTO HTML

Dado que HTML es un lenguaje de marcas, existe una etiqueta (apertura y cierre) especial para señalar que lo que aparece a continuación debe ser analizado y ejecutado por el intérprete de JavaScript en lugar de por el intérprete de HTML: **<script>** ***código JavaScript*** **</script>.** El siguiente código muestra un sencillo ejemplo. Si desea probarlo no tiene más que cortar y pegar el texto en un documento en blanco, guardarlo con el nombre ejemplo.html **¡no se olvide de la extensión!** Y abrirlo con un navegador Web (le recomendamos usar Mozilla Firefox).

```
<html>
<head>
<title>Ejemplo de código JavaScript en el propio documento</title>
<script type="text/javascript">
        alert("Un mensaje de prueba");
</script>
</head>
<body> <p>Un párrafo de texto.</p></body>
</html>
```

Ejemplo 1 – Código Javascript embebido en HTML

La secuencia de resultados se puede ver en las capturas de imagen siguientes. En la izquierda aparece la ventana modal con el texto “Un mensaje de prueba” que es el resultado de interpretar la función Javascript alert. Observe que esta función se ejecuta incluso antes de que el navegador muestre el “body” de la página HTML. Cuando el usuario pulsa el botón Aceptar, el navegador termina de mostrar el resto de la página.

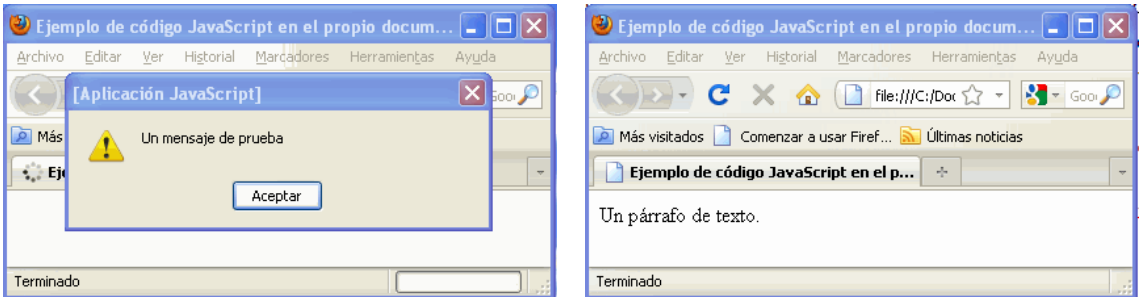


Ilustración 2 – Ejecución de código JavaScript y HTML

Se recomienda este sencillo método cuando la simplicidad del código no impide la legibilidad del mismo. Es importante señalar que el lugar del fichero html en el que se ponen las etiquetas de script, determina en qué momento de la interpretación de la página Web se ejecuta dicho código Javascript. En el ejemplo anterior se ha observado que el código Javascript se ha ejecutado antes que el navegador haya mostrado el cuerpo de la página Web (el mensaje “Un párrafo de texto”).

Además hay una variante de este método que permite incluir código Javascript entremezclado con código HTML. No es un método recomendado porque se pierde mucha legibilidad y su uso se limita a establecer la respuesta Javascript a determinados eventos, como por ejemplo hacer clic sobre un elemento de la página Web. En estos casos, lo habitual es definir funciones Javascript de usuario en un fichero con el segundo método que se describe a continuación y enlazar los eventos con las funciones. El detalle de este mecanismo se estudiará más adelante.

1.3.2 - INCLUIR JAVASCRIPT EN UN FICHERO EXTERNO

Cuando se desea separar el código Javascript del código HTML, con objeto de aumentar la legibilidad y facilitar la reutilización del mismo en otras páginas HTML, se

utiliza un método que permite definir las instrucciones Javascript en un fichero externo al propio fichero HTML. En este caso se debe señalar en el código HTML la ubicación relativa del fichero Javascript respecto al fichero HTML para que el navegador pueda descargar ambos ficheros desde el servidor Web y este localice el código adecuadamente. Veamos un sencillo ejemplo:

Archivo JavaScript: micodigo.js

```
alert("Esto es una alerta escrita en JavaScript");
```

Documento HTML: ejemplo.html

```
<html>
<head>
<title>Código JavaScript almacenado en fichero externo</title>
<script type="text/javascript" src="./micodigo.js"></script>
</head>
<body>
        <p>Esto es un párrafo en HTML</p>
</body>
</html>
```

Ejemplo 2 – Código Javascript en fichero externo

En ambos casos volvemos a tener ficheros de texto plano con dos extensiones diferentes. La primera **js** para indicar que se trata de código Javascript y la segunda **html** para indicar que se trata de código HTML. Observe que el atributo *src* de la etiqueta <script> está indicando la ruta relativa y el nombre en la que se encuentra el fichero **.js** (en este caso ocupan la misma carpeta). Para ejecutar este ejemplo tiene que crear dos ficheros de texto plano independientes y abrir con un navegador el que tiene extensión html.

1.4 - EDITORES HTML Y JAVASCRIPT

Tal y como se ha señalado, el código HTML y el código Javascript son texto plano (sin formato). Por ello se puede utilizar para su edición desde un sencillo editor de texto como el bloc de notas en los entornos Windows o el editor de texto en los entornos Linux hasta un complejo software de diseño de páginas Web tipo *Dreamweaver* o similar (activando la vista de código fuente). En el punto intermedio

de ambas opciones están los editores de sintaxis resaltada que son editores de texto plano pero que incorporan un reconocedor de sintaxis basado en la gramática de los lenguajes que soporta.

1.5 - ENTORNOS DE EJECUCIÓN DE JAVASCRIPT

En general el entorno de ejecución de Javascript es un navegador Web. Esta circunstancia hace que en principio y dado que hay navegadores para las plataformas hardware más importantes, Javascript podría considerarse un lenguaje multiplataforma. Lamentablemente esta consideración es teórica porque no sólo hay diferencias entre la interpretación que hacen diferentes navegadores sobre la misma plataforma hardware sino que incluso a veces la diferencia se produce entre el mismo navegador funcionando en plataformas diferentes o incluso entre versiones diferentes del mismo navegador en la misma o en diferentes plataformas. Esta es una de las principales dificultades a las que se enfrenta el desarrollador de aplicaciones Javascript. En la actualidad, los programadores que utilizan Javascript recurren, cuando la complejidad de la aplicación así lo justifica, al uso de frameworks que entre otras numerosas virtudes resuelven en gran medida el problema señalado, asegurando con mayor o menos éxito el denominado "*Javascript Cross Browser*" recurriendo a un sistema que parte de identificar el navegador que se está utilizando y en función del mismo utilizar el código desarrollado específicamente para el mismo.

Existen numerosos framework algunos de los cuales aparecen reseñados en:
http://www.desarrolloweb.com/articulos/listado-distintos-framework-javascript.html
Existe un interesante artículo comparando varios de ellos en:
http://www.maestrosdelweb.com/editorial/comparacion-frameworks-javascript/

Mención especial merece el caso del GWT de Google que permite, entre otras cosas, desarrollar aplicaciones utilizando lenguaje Java y posteriormente compilarlo para generar Javascript con la característica Cross Browser.

Pero Javascript no está ligado exclusivamente a los navegadores Web sino que existen otras aplicaciones informáticas que pueden ser programadas mediante este

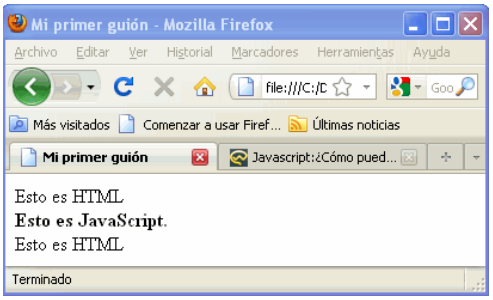
lenguaje o algunos derivados del mismo. Tal es el caso de Adobe Acrobat que permite incluir código Javascript en documentos PDF o el caso de Adobe PhotoShop que permite elaborar algunos scripts para automatizar tareas. Javascript también se puede integrar con el lenguaje Java mediante el paquete javax.script y ha sido fuente de inspiración en el desarrollo del lenguaje Actioscript (Flash y Flex).

1.6 - PRIMERAS APLICACIONES JAVASCRIPT

Tal y como se ha señalado, los programas Javascript se encuentran vinculados a páginas HTML. El navegador reconoce un programa Javascript cuando se encuentra con una etiqueta **<SCRIPT>.** A partir de ese momento será el intérprete Javascript el encargado de interpretar y ejecutar el código hasta la etiqueta **</SCRIPT>.** Para crear un programa Javascript debemos seguir los siguientes pasos:

1. Crearemos una página HTML utilizando cualquier editor, por ejemplo el NotePad ++ o el *Bloc de Notas o un editor Linux de texto plano*. (También podemos utilizar una página ya creada y continuar con el paso siguiente).
2. Insertaremos dentro de la página HTML las etiquetas **<SCRIPT>** y **</SCRIPT>** en el lugar donde se situará el código Javascript.
3. Introduciremos el código Javascript entre dichas etiquetas.
4. Salvaremos el programa poniendo especial cuidado para que el editor mantenga la extensión **.html**
5. Para ejecutar el programa solamente tendremos que abrir el archivo con el navegador.

```
<HTML>
<HEAD><TITLE>El primer programa</TITLE>
</HEAD>
<BODY>
Esto es HTML<br />
<b>
<script type="text/javascript">
        document.write("Esto es JavaScript.");
</SCRIPT>
<br /></b>
Esto es HTML<br/>
</BODY>
</HTML>
```



Ejemplo 3 – El primer programa

La línea central se ha escrito desde un programa JavaScript mediante la instrucción: document.write("Esto es JavaScript."); De hecho, esta es la única instrucción que contiene el programa y su significado es escribir (write) en el documento actual (document) el texto que aparece entre comillas. En realidad document es un objeto que representa el documento HTML actual y write es un método (procedimiento) que se aplica sobre el objeto document y que permite escribir sobre dicho documento. El resultado aparentemente no se distingue de cara al usuario si en lugar de poner el código Javascript se hubiese puesto directamente en HTML el texto. La ventaja de usar Javascript no queda latente en este ejemplo. Veamos otro ejemplo en el que se utiliza una función Javascript que muestra una ventana modal en pantalla.

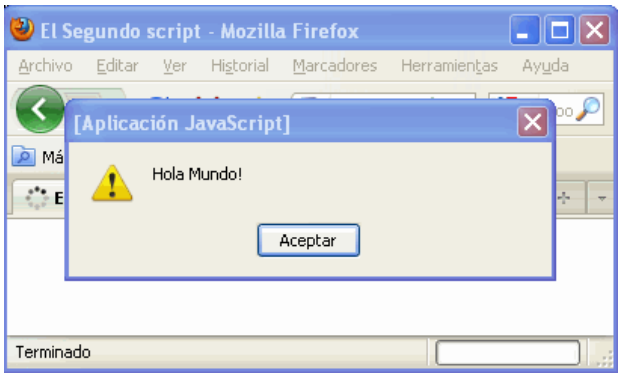
```
<html>
<head>
<title>¡Hola Mundo!</title>
<script type="text/javascript">
        alert("Hola Mundo!");
</script>
</head>
<body>
        <p>Esta página contiene el segundo script</p>
</body>
</html>
```

Ejemplo 4 – Programa "Hola Mundo!"

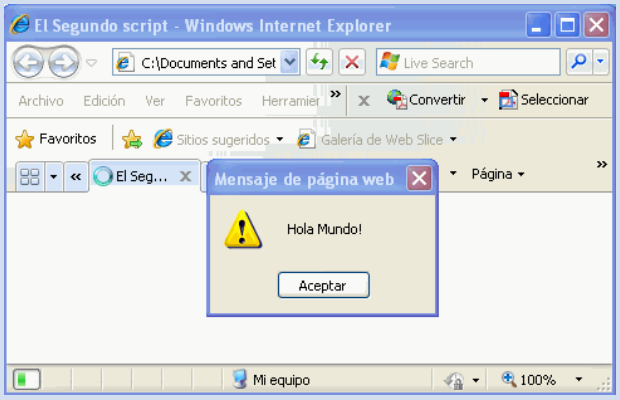
Sólo ejecutando este sencillo ejemplo en diferentes navegadores (Internet Explorer, Mozilla FireFox, Opera, Crome, etc.) podrá observar la diferencias que existen a la hora de interpretarlo. En este sencillo ejemplo se trata de diferencias "estéticas" pero a

medida que avance en la programación de scripts padecerá la falta de consenso entre fabricantes de navegadores a pesar de que existen entidades de normalización.

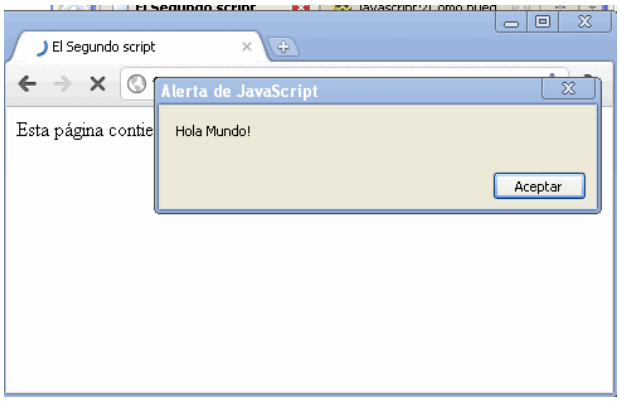
Resultado en Firefox



Resultado en Internet Explorer



Resultado en Google Chrome



Interpretar el código no debería ser muy complicado si se conoce HTML y dado que se ha optado por embeber el código Javascript, este se ejecutará incluso antes de que se muestre en pantalla el texto del body de la página Web porque si observa bien, el script Javascript aparece en el <head> que es la parte que se ejecuta antes de mostrar el <body> en el navegador.

Es importante comprobar que la configuración de seguridad del navegador no impide la ejecución de scripts ya que en algunos casos (por ejemplo, Internet Explorer) la configuración por defecto tiene bloqueada la ejecución de script para proteger el equipo de operaciones susceptibles de ser inseguras. Todos los navegadores avisan de forma diversa sobre esta circunstancia y permite autorizar puntualmente la ejecución de un script o bien establecer por defecto la autorización.

1.7 - COMENTARIOS QUE FACILITAN LA LECTURA

Un programa de ordenador escrito en un lenguaje de programación debe ser interpretado por una maquina. Sin embargo es habitual que el propio programador o cualquier otro acceda al código para modificarlo, corregirlo o simplemente entender su función. Con objeto de que esta lectura sea lo más clarificadora posible es costumbre obligada incorporar comentarios a las instrucciones que constituyen el programa de ordenador, dirigidas a un potencial lector humano. Estos comentarios no tienen efecto sobre el código pero clarifican mucho su comprensión.

Para incluir comentarios en el código Javascript y que estos no interfieran en la labor del intérprete de Javascript ni en el intérprete de HTML hay que ceñirse a una sintaxis específica.

Los comentarios pueden ser:

- **De una línea**. Se indican mediante dos barras inclinadas a la derecha (//). Todo lo que aparezca detrás de dichas barras hasta el final de la línea será considerado comentario y, por tanto, será ignorado por el navegador.

Ejemplo:

```
// Esto es un comentario
document.write("Esto es JavaScript."); //otro comentario
```

- **De varias líneas**. En este caso deberemos indicar el comienzo del comentario mediante los caracteres barra inclinada a la derecha y asterisco (/*). También indicaremos el final del comentario, en este caso mediante la secuencia de

caracteres inversa: asterisco y barra inclinada a la derecha (*/), tal como podemos observar en el siguiente fragmento de código:

```
/* Este es un ejemplo de comentario
de varias  líneas */
```

No obstante y dado que aún hay sistemas informáticos que tienen instalado, por cuestiones de capacidad de proceso, navegadores antiguos es necesario proteger los comentarios para que esos navegadores no lancen un error al encontrar un comentario.

Un comentario HTML tiene la siguiente sintaxis:

```
<!-- aquí va el comentario -->
```

El comentario empieza con los caracteres: <!-- y termina con los caracteres: -->

Para ocultar Javascript a los navegadores antiguos escribiremos los comentarios como se expone a continuación:

```
<SCRIPT LANGUAGE="JavaScript">
<!-- Ocultar guión para navegadores
Aquí va el código JavaScript
// Fin de ocultación del guión -->
</SCRIPT>
```

La última línea de comentario empieza con //, indicando que es un comentario Javascript, y termina con -->, que finaliza un comentario HTML. En algunos navegadores antiguos es necesario incluir un comentario Javascript antes de finalizar el comentario HTML. El siguiente ejemplo muestra el uso de comentarios HTML para ocultar guiones Javascript:

```
<HTML>
<HEAD><TITLE>Utilizando comentarios</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!-- Ocultar guión para navegadores antiguos
function vertexto()
{
        document.write("Esto es JavaScript.")
}
// Fin de ocultación del guión -->
</SCRIPT>
</HEAD>
<BODY>
        Esto es HTML<br />
<b>
<SCRIPT LANGUAGE="JavaScript">
<!-- Ocultar guión para navegadores antiguos
        vertexto();
        // Fin de ocultación del guión -->
</SCRIPT>
</b>
        Esto es HTML <br />
</BODY>
</HTML>
```

Ejemplo 5 – Uso de comentarios

1.8 - ELEMENTOS DE UN PROGRAMA JAVASCRIPT

Un programa es un conjunto ordenado de instrucciones diseñado para llevar a cabo una tarea concreta. Los programas están formados por instrucciones que determinan las tareas que se realizarán y el orden en que estas se llevarán a cabo. Tradicionalmente estas instrucciones se catalogan como:

- **Instrucciones declarativas o declaraciones**: declaran, es decir, informan al intérprete sobre los objetos que van a intervenir en el programa. Por ejemplo, las variables, espacios de memoria identificados por un nombre y capaces de almacenar un valor, se suelen declarar antes de ser utilizadas.

- **Instrucciones de entrada/salida**: permiten que el programa reciba/ envíe datos desde/hasta un dispositivo externo, por ejemplo, teclado y pantalla como principales interfaces de comunicación con el usuario.

- **Instrucciones de control**: determinan la secuencia de ejecución del programa, es decir, qué instrucciones se ejecutan y en qué momento.

El siguiente programa permite al usuario introducir una cantidad en pesetas, calcula el importe equivalente en euros y finalmente lo muestra en pantalla.

```
<HTML>
<HEAD><TITLE>Conversor pesetas a euros</TITLE></HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
var ImportePtas;
var ImporteEuros;
ImportePtas=prompt("Introduce el importe en pesetas:",1);
if (ImportePtas > 0)
{
        ImporteEuros = ImportePtas / 166.33;
        alert(ImporteEuros);
}
else
{
        alert("Importe erróneo");
}
</SCRIPT>
</BODY>
</HTML>
```

Ejemplo 6 – Conversor pesetas a euros

Podemos observar que el programa realiza las siguientes tareas:

- La etiqueta `<SCRIPT LANGUAGE="JavaScript">` indica el comienzo del script.

- Las instrucciones: ***var ImportePtas; var ImporteEuros;***

Declaran las variables *ImportePtas* e *ImporteEuros* para que puedan ser utilizadas posteriormente. Las variables son espacios de memoria del ordenador capaces de almacenar datos.

- La instrucción: ***ImportePtas = prompt("Introduce el importe en pesetas: ",1);***

Visualizará el texto "Introduce el importe en pesetas: ", permitiendo al usuario **introducir** un importe. Dicho importe será **asignado** a la variable ***ImportePtas***. El

número 1 representa el valor por defecto que tomará el cuadro de inserción de manera que el usuario puede modificar dicho o valor o simplemente dejarlo intacto. La ventana modal que aparece como consecuencia de la instrucción prompt muestra dos botones: *Aceptar* y *Cancelar*. Si el usuario pulsa *Aceptar*, la instrucción almacena en la variable ImportePtas el valor que tenga en ese momento el cuadro de texto. Si pulsa *Cancelar* el valor que se asignará a dicha variable será null.

- La instrucción ***if (ImportePtas > 0)*** es una **instrucción de control**. Comprueba el resultado de la condición ***(ImportePtas > 0)*** y si es verdadero ejecutará las instrucciones que se encuentran encerradas entre las llaves que aparecen a continuación. En caso contrario ejecutará las instrucciones que siguen a la cláusula **else**.

- La instrucción: ***ImporteEuros = ImportePtas/166.33;*** **Asigna** a la variable ***ImporteEuros*** el resultado de dividir el valor de la variable ***ImportePtas*** entre 166.33.

- Las instrucciones: ***alert(ImporteEuros); y alert("Importe erróneo");***

Visualizan el contenido de la variable ***ImporteEuros***, o el texto "Importe erróneo", respectivamente, dependiendo de cuál haya sido el resultado de evaluar la condición ***(ImportePtas > 0)***.

- Los caracteres **//** sirven para escribir **comentarios** en el programa. Estos comentarios no se ejecutan. Su utilidad es servir como de ayuda al programador al documentar el programa.

- La etiqueta **</SCRIPT>** señala el final del script.

1.9 - ENTRADA Y SALIDA EN JAVACRIPT

El esquema tradicional empleado en computación responde a un modelo básico y sencillo en el que unos datos de entrada son procesados en base a un algoritmo programado con un lenguaje de programación que los somete a diversos cálculos para generar unos datos de salida. En este modelo la entrada y la salida son fundamentales

porque permiten al usuario proporcionar los datos de partida y conocer los resultados del mismo.

A lo largo de estas primeras unidades utilizaremos con frecuencia instrucciones para visualizar o introducir datos. Por ello vamos a anticipar en este punto algunas características de formato y funcionamiento de estas instrucciones.

1.9.1 -INSTRUCCIONES BÁSICAS DE SALIDA

- La instrucción ***document.write("mensaje")*** permite escribir un texto en el documento actual. Ya hemos visto un ejemplo de utilización en un epígrafe anterior.

- La instrucción ***alert*** se emplea para mostrar mensajes o visualizar datos. Esta instrucción abre una ventana y muestra en ella el mensaje o texto especificado en la llamada. También incluye un botón de *Aceptar* que sirve para cerrar la ventana.

1.9.2 -INSTRUCCIONES BÁSICAS DE ENTRADA

La instrucción prompt muestra una ventana de diálogo que incluye un texto o mensaje, un área de entrada de datos y dos botones: uno de *Aceptar* y otro de *Cancelar*. Su sintaxis es:

```
prompt("mensaje", valorpordefecto);
```

Esta instrucción retorna un valor que normalmente se asignará a una variable según el formato:

```
variable = prompt("mensaje", valorpordefecto);
```

Donde *mensaje:* es un texto o mensaje que queremos que aparezca. *valor_por_defecto*: es una cadena o un número que se asumirá por defecto en el caso de que no se introduzca ningún dato y se pulse *Aceptar*.

Descripción: muestra una ventana de dialogo y espera a recibir una entrada de datos por parte del usuario. La ventana se cierra pulsando uno de los dos botones: *Aceptar*: acepta la entrada del usuario (o la entrada por defecto si esta no ha sido modificada). *Cancelar:* cancela la entrada que pudiese haber introducido el usuario y asume como entrada el valor null.

Si no se especifica ningún valor en valor_por_defecto se asumirá un valor indefinido (<undefined>). Es conveniente especificar siempre un valor por defecto, aunque sea 0 ó "" (cadena vacía). En el ejemplo anterior, la instrucción:

```
ImportePtas = prompt("Introduce el importe en pesetas:",1);
```

Dará como resultado la aparición de una ventana. El navegador esperará a que el usuario introduzca un valor o acepte el valor por defecto y lo asignará a la variable *ImportePtas* tal como se indica al principio de la instrucción.