JavaScript es un lenguaje de programación que permite la ejecución de scripts en el lado cliente de nuestro sistema Web. Apareció en 1995, soportado por el navegador Netscape Navigator 2, e Internet Explorer empezó a tolerarlo a partir de IE3.

¿Qué se puede hacer con Javascript?

- Insertar o modificar contenido de forma dinámica en una página web. Se puede modificar tanto el contenido HTML como las propiedades CSS.
- Recoger información del navegador y equipo cliente.
- Reaccionar a eventos generados en el navegador
- Comunicarse con el servidor sin necesidad de recargar la página Web.

Toda secuencia de comandos se encuentra dentro un bloque `<script>` insertado en el documento HTML:

- En la sección <body>, en cuyo caso se ejecuta el script en cuanto se cargue el código.
- En la sección <head>, que hará que el script se ejecute antes de que el navegador procese parte alguna del código HTML.

Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
       <title> Ejemplo </title>
</head>
<body>
 <h1> Ejemplo de JavaScript.</h1>
 <script type=”text/javascript”>
       alert(“Esto es una ventana emergente con mensaje.”);
 </script>
</body>
</html>
```

Si se mostrara en un navegador el código anterior, el script muestra un mensaje por pantalla cuya ventana se cerrará al pulsar el botón “Aceptar”, que viene por defecto en la función `alert`.

JavaScript reconoce como fin de sentencia el salto de línea o el punto y coma , lo que hace opcional este elemento tan típico de C o Java.

Vamos a comentar algunas nociones básicas de JavaScript, que deberán ser complementadas con el trabajo autónomo en prácticas y la búsqueda de bibliografía, abundante por cierto tanto en la red como en la biblioteca del centro.

Declaración de variables

Para crear una variable en JavaScript se utiliza la palabra clave `var` seguida del nombre de la variable. Hay que destacar que JavaScript, al igual que C, es sensible a las mayúsculas. Por supuesto, es posible inicializarlas a la vez que se declaran, pero nótese que no es un lenguaje fuertemente tipado, que incluso dispone de operador de concatenación de texto (el símbolo +)

```
var ejemplo=”Texto de ejemplo”;
var concatenacion=ejemplo+” concatenado”;
```

Veamos un ejemplo algo más complejo, en el que se muestra la fecha actual:

```
<!DOCTYPE html>
<html>
<head>
       <title> Ejemplo JavaScript 2 </title>
</head>
<body>
 <h1> Ejemplo de JavaScript - Fecha.</h1>
 <p>
 <script type=”text/javascript”>
       var fechaActual = new Date();  // 1
       var mensaje = “Hoy es “;       // 2
       document.write (mensaje + fechaActual.toDateString()); // 3
 </script>
 </p>
</body>
</html>
```

Lo más reseñable es que en la línea etiquetada como // 3, JavaScript incluye en el HTML, en la posición del script, la cadena de texto formada por la concatenación del mensaje “Hoy es “ con el string de la fecha actual, obtenido mediante el método `toDateString()`. Fíjese que esto ya nos indica que JavaScript es orientado a objetos.

Funciones

Las funciones JavaScript suelen declararse en la sección <head> de la página, y luego son invocadas desde dentro del contenido <body>, por ejemplo

```
<!DOCTYPE html>
<html>
<head>
       <title> Ejemplo JavaScript 2 </title>
<script>
       function getMensajeConFecha(mensaje) {
             var fechaActual = new Date();
             return mensaje + fechaActual.toDateString();
       }
</head>
<body>
 <h1> Ejemplo de JavaScript - Fecha.</h1>
 <p>
 <script type=”text/javascript”>
     // Llamamos a la función definida en el head
       document.write (getMensajeFecha(“Hoy es “));
 </script>
 </p>
</body>
</html>
```

Otra opción, mucho más elegante, es guardar el código de las funciones en un archivo externo, y referenciarlo en la sección head:

```
<script src=”funciones.js”></script>
```

HTML Dinámico y Javascript

A finales de los 90, apareció el concepto de HTML dinámico, una nueva forma de pensar en el diseño y la ejecución de una página Web. Con el modelo de objetos de un documento HTML (DOM), cada uno de los elementos de la página son objetos jerarquizados que pueden ser accedidos, y sus propiedades modificadas, mediante el uso de JavaScript.

Antes de poder manipular un objeto en la página Web, es fundamental poder identificarlo de forma unívoca, y para ello lo mejor es usar el atributo `id` que ya vimos en el apartado de CSS alguna de sus utilidades.

```
<h1 id="encabezado1"> Bienvenidos </h1>
```

Una vez tenemos con id el elemento, se puede obtener el objeto de dicho elemento con JavaScript utilizando el método `document.getElementById()`. Básicamente, `document` es un objeto que representa al documento HTML completo, es una variable global que tiene el navegador y siempre está disponible.

En nuestro caso, podríamos modificar el contenido del elemento h1 anterior con las siguientes líneas:

```
var objetoTitulo = document.getElementById("encabezado1");
objetoTitulo.innerHTML = "Bienvenidos todos";
```

Todos los objetos HTML comparten algunos atributos comunes, como los mostrados en la siguiente tabla:

Propiedad	Descripción
`className`	Permite recuperar o establecer el atributo `class`, utilizado para aplicar estilos CSS.
`innerHTML`	Permite leer o cambiar el HTML del interior de un elemento.
`parentElement`	Proporciona el objeto HTML en el que está contenido el elemento.
`style`	Aúna todos los atributos CSS del elemento HTML. Devuelve un objeto con las propiedades CSS como atributo.
`tagName`	Proporciona el nombre del elemento HTML (p.ej. si es un `<img>` devuelve `img`)

El uso del DOM de HTML5 permite conseguir animaciones y efectos muy conseguidos con simples rutinas.

Eventos

Los eventos son notificaciones que envía un elemento HTML cuando ocurre algo. Por ejemplo, JavaScript asigna a todo elemento <a> un evento llamado `onmouseover`. Cuando un visitante pasa el puntero del ratón sobre dicho elemento, se inicia el evento:

```
<p> En un párrafo <a href="#" onmouseover="alert('Hey!')">ten cuidado</a></p>
```

En la siguiente tabla mostramos los eventos más utilizados y los elementos HTML a los que se aplican:

Evento	Descripcióin	Elementos
onClick	Se inicia al hacer click	Casi todos
onMouseOver	Se inicia al pasar el puntero sobre el elemento	Casi todos
onMouseOut	Se inicia al alejar el puntero del elemento	Casi todos
onKeyUp	Se inicia al soltar una tecla pulsada	<select>,<input>, <textarea>,<a> <button>
onFocus	Se inicia cuando un control recibe el foco	<select>,<input>, <textarea>,<a> <button>
onChange	Se inicia cuando cambia un valor en un control de entrada	<select>,<input type=”text”>, <textarea>
onLoad	Se inicia cuando el navegador termina de cargar una página o elemento	<img>, <body>