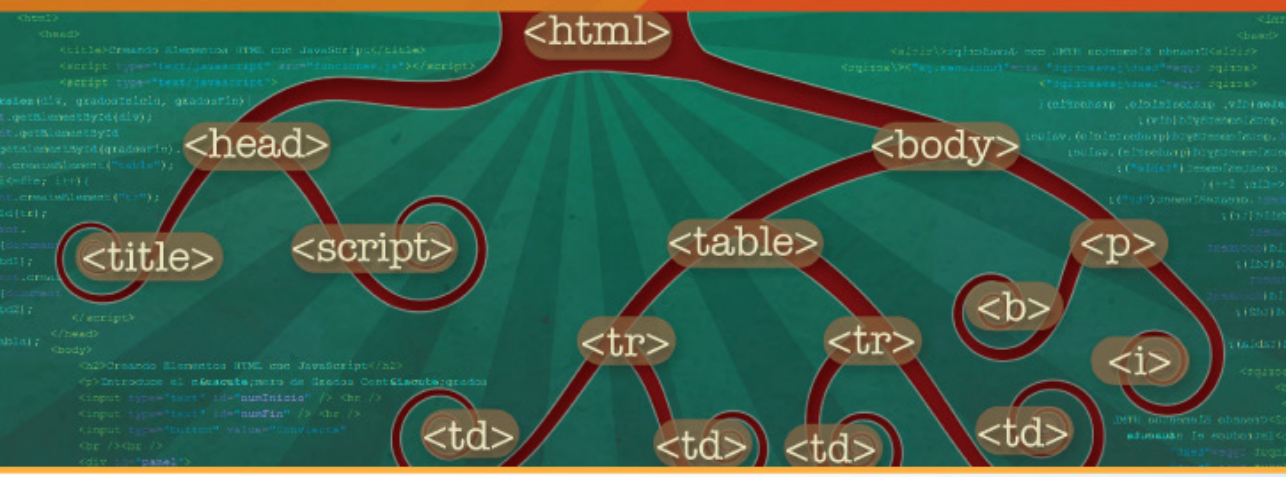
Programación de Web Dinámico

Carlos Roberto Jaimez González

Obra ganadora Concurso 2014 para publicación de libros de texto y materiales de apoyo a la Docencia



10 años
2005 – 2015
Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA
Unidad Cuajimalpa

Carlos Roberto Jaimez González

Programación de Web Dinámico

Esta investigación fue dictaminada por pares académicos

Clasificación Dewey: 005.13 J35
Clasificación LC: QA76.7 J35
Jaimez González, Carlos Roberto
Programación de web dinámico / Carlos Roberto Jaimez -- México : UAM, Unidad Cuajimalpa, c2015.
88 p. : il., tablas, col. ; 24 cm. -- (Una década de la Unidad Cuajimalpa de la Universidad Autónoma Metropolitana)
ISBN de la Colección Una Década: 978-607-28-0452-4
ISBN de este libro: 978-607-28-0470-8
1. Lenguajes de programación (Computadoras) – Libros de texto 2. Desarrollo de sitios web – Libros de texto 3. JavaScript (Lenguaje de programación para computadora) – Libros de texto 4. Document Object Model (Tecnología para el desarrollo de sitios web) – Libros de texto 5. HTML (Lenguaje de marcado de hipertexto) – Libros de texto 6. Universidad Autónoma Metropolitana – Unidad Cuajimalpa – Planes de estudio 7. Planes de estudio universitario - México

D.R. © 2015 UNIVERSIDAD AUTÓNOMA METROPOLITANA
Universidad Autónoma Metropolitana Unidad Cuajimalpa
.Avenida Vasco de Quiroga 4871,
Col. Santa Fe Cuajimalpa. Delegación Cuajimalpa de Morelos,
C.P. 05348, México D.F. (Tel.: 5814 6500)
www.cua.uam.mx

ISBN de la Colección Una Década: 978-607-28-0452-4
ISBN de este libro: 978-607-28-0470-8

Diseño de portada: Ricardo López Gómez.
Formación y edición: Juan Carlos Rosas Ramírez.

CONTENIDO

Agradecimientos

Agradezco a los revisores anónimos de este material por sus valiosos comentarios y observaciones. También agradezco a Wulfrano Luna Ramírez por sus sugerencias y comentarios, así como por la labor de señalar puntualmente las modificaciones pertinentes a este texto. Asimismo, expreso mi gratitud a Betzabet García Mendoza por su valiosa opinión y sugerencias acerca del texto en general, pero principalmente por su revisión a los ejemplos y ejercicios contenidos en este material.

La oportunidad de ser profesor universitario me ha permitido crear ejemplos y ejercicios para complementar mis clases, algunos de estos se encuentran en el presente texto. Muchas gracias a todos los alumnos con quienes he compartido el salón de clases en la Universidad Autónoma Metropolitana, ustedes son la principal motivación para la creación de estos materiales.

Gracias a la colaboración de todos ustedes he podido mejorar la calidad y presentación de este trabajo.

Introducción

El presente material didáctico tiene como objetivo apoyar la enseñanza de la UEA Programación de Web Dinámico de la Licenciatura en Tecnologías y Sistemas de Información de la Universidad Autónoma Metropolitana Unidad Cuajimalpa. El contenido de este trabajo es útil no solo para estudiantes de dicha UEA, sino también para personas con conocimientos sobre creación de páginas web estáticas que utilicen el lenguaje de marcado de hipertexto (HTML) y las hojas de estilo en cascada (CSS), quienes estén interesadas en un curso introductorio acerca del desarrollo de páginas web dinámicas e interactivas utilizando el lenguaje de programación JavaScript y el modelo de objetos de documento (DOM).

Este material de apoyo aborda diversos temas que cubren parte del contenido sintético de la UEA Programación de Web Dinámico. Específicamente abarca los temas relacionados a la programación web dinámica del lado del cliente en un navegador web.

La información que se proporciona en este material ayudará al estudiante a entender el funcionamiento y desarrollar sus propios sitios web dinámicos e interactivos con JavaScript. Este material también le permitirá adquirir los conocimientos y habilidades necesarias para posteriormente incursionar en el desarrollo de aplicaciones web dinámicas del lado del servidor con tecnologías como JSP o PHP.

La organización de este título se distribuyó en cinco capítulos, bibliografía de consulta y apoyo, además de un glosario de términos. En cada uno de los capítulos se brinda una explicación de los temas que serán cubiertos; ejemplos con los documentos HTML que contienen código JavaScript, los cuales se muestran completos y explicados detalladamente; figuras con las páginas web como serán visualizadas en un navegador web, así como ejercicios para reforzar lo aprendido en los temas del capítulo.

A su vez, los capítulos se subdividen de la siguiente manera. En el capítulo uno se presenta una introducción al lenguaje de programación JavaScript, la estructura básica de un documento HTML traducido a un árbol de elementos DOM, así como los diferentes tipos de objetos que se manejan en un documento HTML. El capítulo dos explica las diferentes formas de encontrar elementos HTML en una página web. El capítulo tres describe cómo manejar los eventos que ocurren en un documento HTML y responder a estos mediante la ejecución de código JavaScript. La creación y eliminación de elementos en un documento HTML a través de DOM con JavaScript se muestra en el capítulo cuatro. Finalmente, en el capítulo cinco se expone la manera en que se pueden cambiar los estilos de los elementos de un documento HTML para modificar la apariencia de una página web.

Capítulo 1. Introducción a JavaScript y DOM

En este capítulo se ofrece una introducción a JavaScript, lenguaje *script* basado en objetos y que está diseñado específicamente para hacer que las páginas web sean dinámicas e interactivas. Además, se presenta el Modelo de Objetos de Documento (Document Object Model, DOM, por sus siglas en inglés), mismo que traduce la estructura de una página web a un árbol de objetos cuando esta es cargada en un navegador web. Al utilizar JavaScript y DOM será posible cambiar cualquier elemento de una página web a través de los métodos y atributos de los objetos. En este capítulo también se proporcionan ejemplos y ejercicios para ilustrar el uso de objetos, métodos y atributos en una página web con JavaScript, así como para identificar los objetos en el árbol generado con DOM en una página web.

Los objetivos que deben cumplirse al final de este capítulo son:

- Conocer las características más importantes del lenguaje JavaScript.
- Identificar las diferencias entre objetos, métodos y atributos.
- Utilizar el objeto *document* para escribir dinámicamente en una página web.
- Conocer qué es DOM y cómo transforma una página web a un árbol de objetos.
- Identificar los diferentes tipos de objetos en un árbol generado con DOM.

El lenguaje JavaScript

JavaScript es un lenguaje *script* basado en objetos, diseñado específicamente para hacer que las páginas web sean dinámicas e interactivas. JavaScript es un lenguaje para hacer programación web dinámica del lado del cliente.

Un lenguaje *script* es un lenguaje de programación interpretado que requiere de un intérprete, quien traduzca las sentencias escritas en el lenguaje

a código máquina cada vez que el programa es ejecutado. JavaScript es interpretado por navegadores web, los cuales representan a los clientes.

El lenguaje JavaScript está basado en objetos, ya que proporciona una implementación del DOM, modelo que traduce la estructura de un documento HTML a un árbol de objetos cuando este es cargado en un navegador web. Cada objeto tiene métodos y atributos que pueden ser invocados desde código JavaScript para su manipulación con el propósito de cambiar cualquier elemento del documento HTML.

JavaScript puede ser incrustado directamente en el código de un documento HTML o mediante archivos separados, con el fin de que un navegador web los interprete y los ejecute cuando sea requerido. Los archivos creados con JavaScript son en texto plano, sin formato, que se almacenan con la extensión *.js* y están asociados a documentos HTML. El código JavaScript puede ser interpretado por cualquier navegador web moderno, como Mozilla Firefox, Google Chrome, Opera, Safari, Internet Explorer, entre otros.

En cada ejemplo que se presente en este material se proporcionarán los listados correspondientes a los documentos HTML, con extensión *.html*, los cuales contienen las etiquetas HTML e incluyen los segmentos de código o funciones JavaScript que se utilizan, las cuales serán incrustadas directamente en los documentos HTML.

Se puede emplear cualquier editor de texto para crear los documentos HTML de los ejemplos, así como cualquier navegador web para visualizarlos. En las figuras que se muestran en los ejemplos y ejercicios se utilizó Mozilla Firefox como navegador web. En todos los listados de los ejemplos se presentan los números de línea en la columna de la izquierda para facilitar su explicación.

Objetos, atributos y métodos

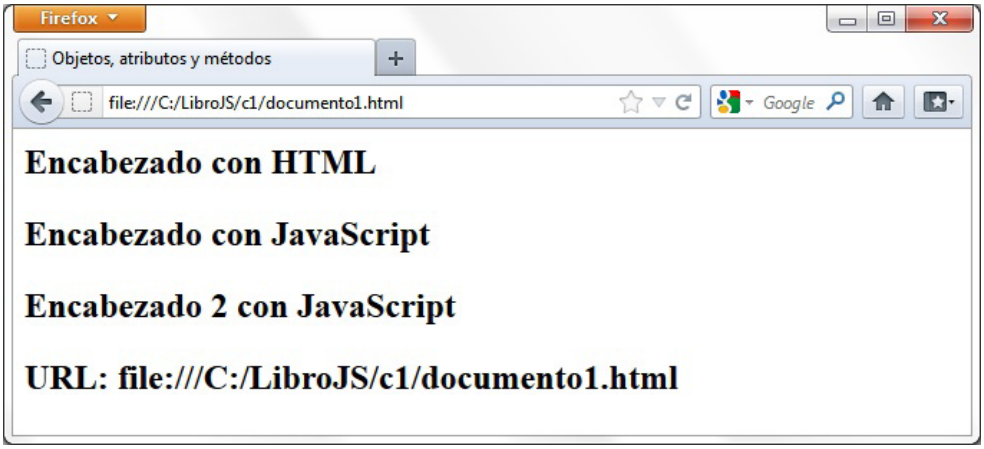
Un objeto en JavaScript es un elemento que proviene de un documento HTML a partir de que el documento se cargó en un navegador web. El objeto contiene atributos que describen sus características y métodos, los cuales representan el comportamiento o las acciones que el objeto puede ejecutar.

Cuando se carga un documento HTML en un navegador web, este es representado por un objeto `document` en JavaScript, a partir del cual se puede tener acceso al resto de la página web. Este objeto posee atributos y métodos para ejecutar acciones sobre el documento HTML al que representa.

Ejemplo: Atributos y métodos del objeto document

Este ejemplo muestra el uso de un atributo y un método del objeto `document`. En la figura 1.1 se muestra una página web con cuatro encabezados HTML de segundo nivel: el primero de ellos fue escrito directamente con HTML y los tres restantes se escribieron dinámicamente a través de JavaScript con el método `write()` del objeto `document`.

Figura 1.1. Página web con ejecución de código JavaScript



En el listado 1.1 se muestra el archivo *documento1.html* que es utilizado para visualizar la página web que se presentó en la figura 1.1, en el cual se utiliza un método y un atributo del objeto `document`.

Listado 1.1. Archivo documento1.html visualizado en la figura 1.1

```
1   <html>
2       <head>
3           <title>Objetos, atributos y m&eacute;todos</title>
4       </head>
5       <body>
6           <h2>Encabezado con HTML</h2>
7           <script type="text/javascript">
8              document.write("<h2>Encabezado con JavaScript</h2>");
9              document.write("<h2>Encabezado 2 con JavaScript</h2>");
10             document.write("<h2>URL: " + document.URL + "</h2>");
11          </script>
12      </body>
13  </html>
```

El listado 1.1 crea un documento HTML con un encabezado y un segmento de código JavaScript. En la línea 6 del listado, se muestra un encabezado de segundo nivel delimitado por la etiqueta inicial `<h2>` y la etiqueta final `</h2>`. En las líneas 7 a 11 se encuentra un segmento de código JavaScript que está delimitado por la etiqueta inicial `<script>` y la etiqueta final `</script>`, donde se tiene el atributo `type="text/javascript"` para indicar que el segmento de código corresponde al lenguaje JavaScript. Cuando el navegador web carga este documento HTML y encuentra el segmento de JavaScript en las líneas 7 a 11, ejecuta sus tres sentencias. La primera sentencia JavaScript está en la línea 8 y representa la invocación del método `write()` del objeto `document`, el cual se encarga de escribir en la página web la cadena de texto que se encuentra como parámetro; en este caso escribe un encabezado de segundo nivel con el texto `Encabezado con JavaScript`. La segunda sentencia también hace uso del método `write()` del objeto `document` para escribir en la página web otro encabezado de segundo nivel, pero ahora con el texto `Encabezado 2 con JavaScript`. La tercera sentencia, que se encuentra en la línea 10, nuevamente hace uso método `write()` del objeto `document` para escribir en la página web otro encabezado de segundo nivel, pero ahora hace una concatenación de cadenas de texto utilizando el atributo `URL` del objeto `document`, el cual regresa el URL del documento HTML que se está visualizando. Como se observa en la figura 1.1, el resultado de la ejecución de estas sentencias JavaScript es el que se describió en este listado: un encabezado escrito directamente con HTML y tres generados dinámicamente mediante la ejecución de sentencias JavaScript.

Ejercicio: Tabla creada dinámicamente con el objeto document

Escribe un documento HTML con el nombre *documento2.html*. El documento debe visualizarse en el navegador web como se muestra en la figura 1.2. Toma como base el ejemplo anterior, elimina las tres sentencias de las líneas 8 a 10 y añade las sentencias JavaScript necesarias para escribir dinámicamente la tabla de la figura 1.2. En la tabla se hace uso de dos atributos del objeto `document`: el atributo `URL`, el cual regresa el URL donde se encuentra el documento HTML y el atributo `title`, el cual regresa el título de la página web que se encuentra delimitado por la etiqueta inicial `<title>` y la etiqueta final `</title>`.

Figura 1.2. Página web con tabla creada dinámicamente mediante JavaScript

El Document Object Model (DOM)

DOM es un modelo que traduce la estructura de un documento HTML a un árbol de objetos cuando este es cargado en un navegador web. JavaScript puede acceder y cambiar todos los elementos de un documento HTML a través de este modelo.

Ejemplo: Documento HTML y su árbol de objetos

Este ejemplo muestra un documento HTML y su árbol de objetos asociado, el cual es generado mediante DOM. La figura 1.3 muestra un documento HTML visualizado en un navegador web que contiene un encabezado de segundo nivel, un párrafo y un hipervínculo. El listado 1.2 muestra el archivo *arbolObjetos.html*, utilizado para visualizar la página web de la figura 1.3. Finalmente, la figura 1.4 muestra el árbol de objetos DOM asociado con este documento HTML.

Figura 1.3. Documento HTML con encabezado, párrafo e hipervínculo

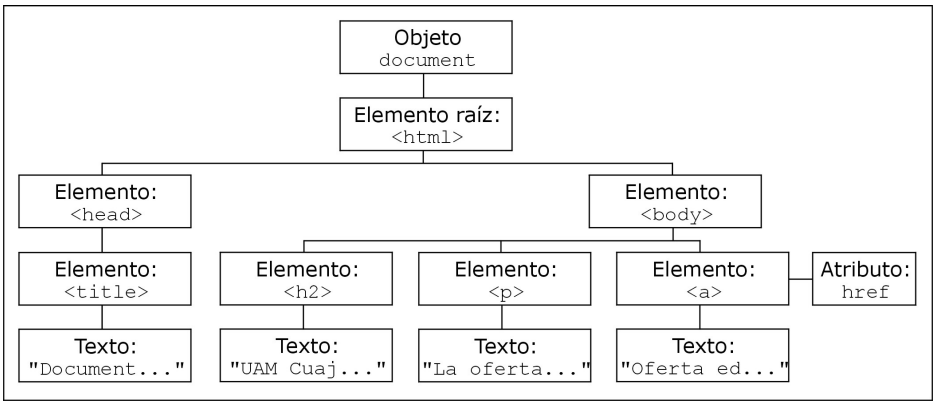
En el listado 1.2 se muestra el archivo *arbolObjetos.html*, donde se observa un encabezado <h2>, un párrafo <p> y un hipervínculo <a>.

Listado 1.2. Archivo *arbolObjetos.html* visualizado en la figura 1.3

```
<html>
    <head>
        <title>Document Object Model</title>
    </head>
    <body>
        <h2>UAM Cuajimalpa</h2>
        <p>La oferta educativa de la Universidad
           Aut&oacute;noma Metropolitana, Unidad
           Cuajimalpa puede consultarse en el URL:</p>
        <a href="http://www.cua.uam.mx">Oferta Educativa</a>
    </body>
</html>
```

En la figura 1.4 se presenta el árbol de objetos generado con DOM una vez que el documento HTML ha sido cargado en el navegador web. Puede observarse que la jerarquía de los objetos generados coincide con la jerarquía de las etiquetas contenidas en el documento HTML. El nodo raíz es el objeto `document` que representa al documento HTML completo; le sigue el elemento raíz <html>, del cual se tienen dos nodos: el elemento <head> y el elemento <body>. Del elemento <head> se tiene como nodo hijo el elemento <title>, que a su vez tiene un nodo de texto con la cadena `Document Object Model`, el cual es un nodo hoja. Del elemento <body> se tienen tres objetos hijo: el elemento <h2>, el elemento <p> y el elemento <a>. El elemento <h2> tiene un nodo de texto, el elemento <p> también tiene un nodo de texto y, finalmente, el elemento <a> tiene un nodo de texto y un objeto del tipo atributo para `href`.

Figura 1.4. Árbol de objetos DOM para el documento HTML del listado 1.2

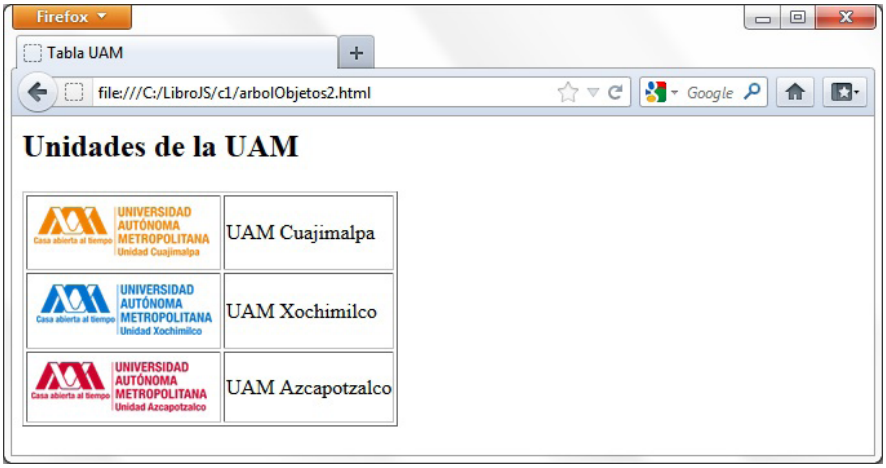


Por medio de este modelo de objetos, JavaScript puede crear HTML dinámicamente; es capaz de añadir, modificar y eliminar elementos y atributos en un documento HTML. Además, con este modelo de objetos, JavaScript puede reaccionar a todos los eventos HTML existentes en una página web, como se verá más adelante.

Ejercicio: Árbol de objetos para un documento HTML con una tabla

Escribe un documento HTML con el nombre *arbolObjetos2.html* y dibuja su representación como árbol de objetos DOM. El documento debe visualizarse en el navegador web como se muestra en la figura 1.5. El documento HTML contiene un encabezado de segundo nivel y una tabla con tres renglones, donde cada renglón contiene celdas con imágenes y texto.

Figura 1.5. Documento HTML con encabezado y tabla con imágenes y texto



Capítulo 2. Acceso a elementos HTML

En este capítulo se presentan las diferentes formas para encontrar elementos HTML en una página web, sea por su identificador, el nombre de su etiqueta HTML, el nombre de su clase o la colección a la que pertenecen. Cada una de las formas que se presentan en las secciones de este capítulo utiliza diferentes métodos del objeto *document*, el cual representa al documento HTML. También se proporcionan varios ejemplos y ejercicios para ilustrar el uso de los métodos y atributos necesarios para encontrar elementos en una página web, así como para modificarlos.

Los objetivos que deben cumplirse al final de este capítulo son:

- Conocer las diferentes formas para encontrar elementos HTML en una página web.
- Utilizar los métodos y atributos correspondientes del objeto *document* para encontrar elementos HTML, sea por identificador, nombre de etiqueta, nombre de clase y la colección a la que pertenecen.
- Utilizar el atributo *innerHTML* para modificar el contenido de cualquier elemento HTML en una página web.
- Utilizar arreglos de elementos HTML para acceder a sus atributos.

Métodos para encontrar objetos

Como se mencionó en el capítulo anterior, con DOM el objeto `document` representa un documento HTML o página web. A través del objeto `document` se puede encontrar y tener acceso a todos los demás objetos o elementos HTML de una página web para manipularlos.

Existen varias formas para encontrar elementos HTML en una página web y manipularlos: por el valor de su atributo `id`, el cual es un identificador único

que se asigna al elemento HTML; por el nombre de su etiqueta HTML; por el valor de su atributo `class`, el cual representa el nombre de la clase a la cual pertenece, y por las colecciones de objetos HTML predeterminadas. En las siguientes secciones se presentarán ejemplos y ejercicios de estas diferentes formas para encontrar elementos HTML en una página web.

Encontrando objetos por su identificador

La forma más sencilla de encontrar un elemento HTML con DOM es utilizando el atributo `id` del elemento, el cual identifica de manera única al elemento en la página web. Para encontrar un elemento por su identificador se utiliza el método `getElementById()` del objeto `document`, el cual recibe como parámetro el valor del atributo `id` del elemento que se desea encontrar. El método `getElementById()` regresa una referencia al elemento HTML si lo encuentra o `null` si no lo encuentra.

Ejemplo: Encontrando párrafos por su identificador

En este ejemplo se realizará un documento HTML para mostrar cómo encontrar elementos HTML por su identificador.

A partir del contenido que se muestra en el listado 2.1, se creará un documento HTML con el nombre *encuentraParrafosPorId.html*; después, se abrirá en un navegador web.

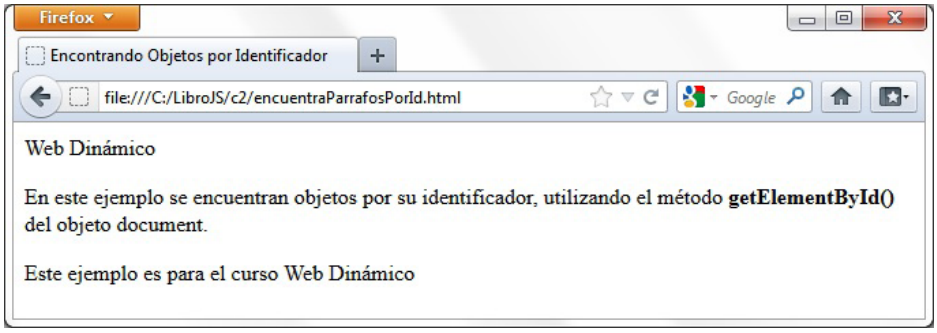
Listado 2.1. Archivo *encuentraParrafosPorId.html*

```
<html>
    <head>
        <title>Encontrando Objetos por Identificador</title>
    <body>
        <p id="curso">Web Din&aacute;mico</p>
        <p>En este ejemplo se encuentran objetos por su
           identificador, utilizando el m&eacute;todo
           <b>getElementById()</b> del objeto document.</p>
        <p id="texto"></p>
        <script type="text/javascript">
            miCurso = document.getElementById("curso");
            miTexto = document.getElementById("texto");
          miTexto.innerHTML = "Este ejemplo es para el curso" +
                                    miCurso.innerHTML;
        </script>
    </body>
</html>
```

El listado 2.1 crea un documento HTML con tres párrafos y un segmento de código JavaScript. En la línea 5 del listado se encuentra el primer párrafo con el texto `Web Din&aacute;mico` y con su atributo `id="curso"`, el cual representa el identificador de ese elemento HTML. En las líneas 6 a 8 se encuentra el segundo párrafo con un fragmento de texto, el cual no tiene identificador. El tercer párrafo se encuentra en la línea 9 con su atributo `id="texto"`, el cual representa el identificador de ese elemento HTML y no tiene contenido entre su etiqueta inicial `<p>` y su etiqueta final `</p>`. Las líneas 10 a 15 contienen sentencias JavaScript delimitadas por la etiqueta inicial `<script>` y la etiqueta final `</script>`. La primera de las sentencias, en la línea 11, encuentra el elemento HTML cuyo identificador es `curso`, mediante la invocación al método `getElementById("curso")` del objeto `document`, y almacena la referencia al elemento en la variable `miCurso`. La línea 12 realiza algo similar a la línea anterior, ya que encuentra el elemento HTML cuyo identificador es `texto`, mediante la invocación al método `getElementById("texto")` del objeto `document`, y almacena la referencia al elemento en la variable `miTexto`. Finalmente, las líneas 13 y 14 concatenan la cadena de texto "`Este ejemplo es para el curso`" con el valor del atributo `innerHTML` del objeto `miCurso`, el cual contiene el HTML que está delimitado entre la etiqueta inicial y final de ese elemento, que en este caso es el texto "`Web Din&aacute;mico`", dando como resultado la cadena "`Este ejemplo es para el curso Web Din&aacute;mico`", la cual es asignada al atributo `innerHTML` del objeto `miTexto` que corresponde al tercer párrafo del documento HTML. Con estas sentencias JavaScript, el resultado es que el tercer párrafo de la página web tendrá como contenido el texto "`Este ejemplo es para el curso Web Din&aacute;mico`". Esto se realizó mediante la recuperación de elementos HTML con el método `getElementById()` y usando el atributo `innerHTML` para modificar el contenido de los elementos.

La figura 2.1 muestra la página web *encuentraParrafosPorId.html* de este ejemplo visualizada en un navegador web, donde se observa que el contenido del tercer párrafo es el texto "`Este ejemplo es para el curso Web Din&aacute;mico`", el cual fue generado dinámicamente, como ya se explicó.

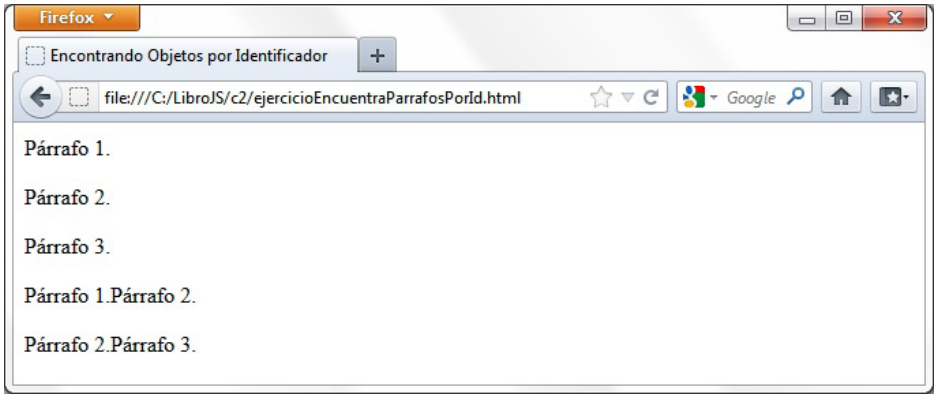
Figura 2.1. Encontrando párrafos por identificador



Ejercicio: Encontrando párrafos y modificando su contenido

Escribe un documento HTML con el nombre *ejercicioEncuentraParrafosPorId.html*. El documento debe visualizarse en el navegador web como se muestra en la figura 2.2. Los tres primeros párrafos tienen el contenido mostrado: Párrafo 1, Párrafo 2 y Párrafo 3. El contenido de los párrafos 4 y 5 es generado a partir del contenido de los tres primeros párrafos: el párrafo 4 es la concatenación del contenido de los párrafos 1 y 2. El párrafo 5 es la concatenación del contenido de los párrafos 2 y 3. Para este ejercicio debes encontrar los párrafos en la página web para realizar lo que se solicita.

Figura 2.2. Página web para encontrar párrafos y modificar su contenido



Encontrando objetos por el nombre de su etiqueta

Otra forma de encontrar elementos HTML con DOM es utilizando el nombre de su etiqueta en la página web. Para encontrar elementos por el nombre de su etiqueta se utiliza el método `getElementsByTagName()` del objeto `document`, el cual recibe como parámetro el nombre de la etiqueta de los elementos que se desea encontrar. El método `getElementsByTagName()` regresa un arreglo con los elementos HTML que tienen la etiqueta especificada si existen o `null` si no los encuentra.

Ejemplo: Encontrando párrafos por el nombre de su etiqueta

En este ejemplo se realizará un documento HTML para mostrar cómo encontrar elementos HTML por el nombre de su etiqueta.

A partir del contenido que se muestra en el listado 2.2, se creará un documento HTML con el nombre *encuentraParrafosPorEtiqueta.html*; después, se abrirá en un navegador web.

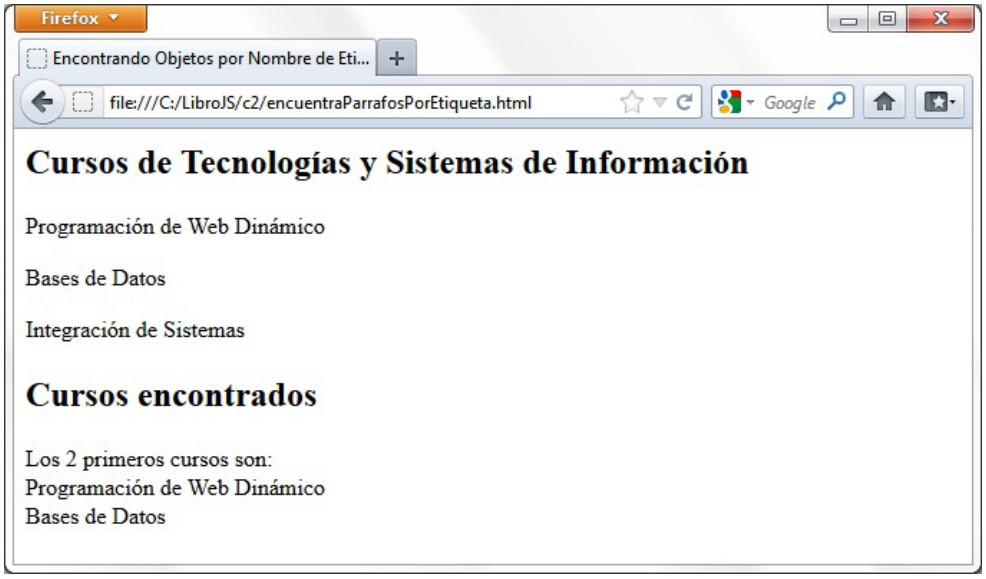
Listado 2.2. Archivo *encuentraParrafosPorEtiqueta.html*

```
<html>
    <head>
        <title>Encontrando Objetos por Nombre de Etiqueta</
title>
    <body>
        <h2>Cursos de Tecnolog&iacute;as
            y Sistemas de Informaci&oacute;n</h2>
        <p>Programaci&oacute;n de Web Din&aacute;mico</p>
        <p>Bases de Datos</p>
        <p>Integraci&oacute;n de Sistemas</p>
        <h2>Cursos encontrados</h2>
        <p id="texto"></p>
        <script type="text/javascript">
            parrafos = document.getElementsByTagName("p");
            miTexto = document.getElementById("texto");
            miTexto.innerHTML = "Los 2 primeros cursos son:" +
                                "<br />" + parrafos[0].innerHTML +
                                 "<br />" + parrafos[1].innerHTML;
        </script>
    </body>
</html>
```

El listado 2.2 crea un documento HTML con cuatro párrafos y un segmento de código JavaScript. En la línea 7 del listado se encuentra el primer párrafo con el texto `Programaci&oacute;n de Web Din&aacute;mico`. En la línea 8 se encuentra el segundo párrafo con el texto `Bases de Datos`. El tercer párrafo se encuentra en la línea 9 con el texto `Integraci&oacute;n de Sistemas`. En la línea 11 se encuentra el cuarto párrafo con su atributo `id="texto"`, el cual representa el identificador de ese elemento HTML y no tiene contenido entre su etiqueta inicial `<p>` y su etiqueta final `</p>`, ya que este párrafo será utilizado para escribir su contenido mediante JavaScript. Las líneas 12 a 18 contienen sentencias JavaScript delimitadas por la etiqueta inicial `<script>` y la etiqueta final `</script>`. La primera de las sentencias, en la línea 13, se encarga de encontrar los elementos HTML cuyo nombre de etiqueta sea `p`, mediante la invocación al método `getElementsByTagName("p")` del objeto `document`, y almacena la referencia al arreglo de los elementos encontrados en la variable `parrafos`. La línea 14 encuentra el elemento HTML cuyo identificador es `texto`, mediante la invocación al método `getElementById("texto")` del objeto `document`, y almacena la referencia al elemento en la variable `miTexto`. Finalmente, las líneas 15 a 17 concatenan la cadena de texto `"Los 2 primeros cursos son:"` con el valor del atributo `innerHTML` de los dos primeros elementos del arreglo de `parrafos`, los cuales son accedidos a través de su índice: `parrafos[0]` y `parrafos[1]`, y corresponden al primer y segundo párrafos encontrados en la página web, respectivamente. La cadena resultante es asignada al atributo `innerHTML` del objeto `miTexto`, el cual corresponde al cuarto párrafo del documento HTML. Con estas sentencias JavaScript, el resultado de este ejemplo es que el cuarto párrafo de la página web tendrá como contenido el texto `"Los 2 primeros cursos son: <br /> Programaci&oacute;n de Web Din&aacute;mico <br /> Bases de Datos"`. Esto se realiza mediante la recuperación de elementos HTML con el método `getElementsByTagName()` y usando el atributo `innerHTML` para modificar el contenido de los elementos.

La figura 2.3 muestra la página web *encuentraParrafosPorEtiqueta.html* de este ejemplo visualizada en un navegador web, donde se observa que el contenido del cuarto párrafo es el texto `"Los 2 primeros cursos son: <br /> Programaci&oacute;n de Web Din&aacute;mico <br /> Bases de Datos"` el cual fue generado dinámicamente, como ya se explicó.

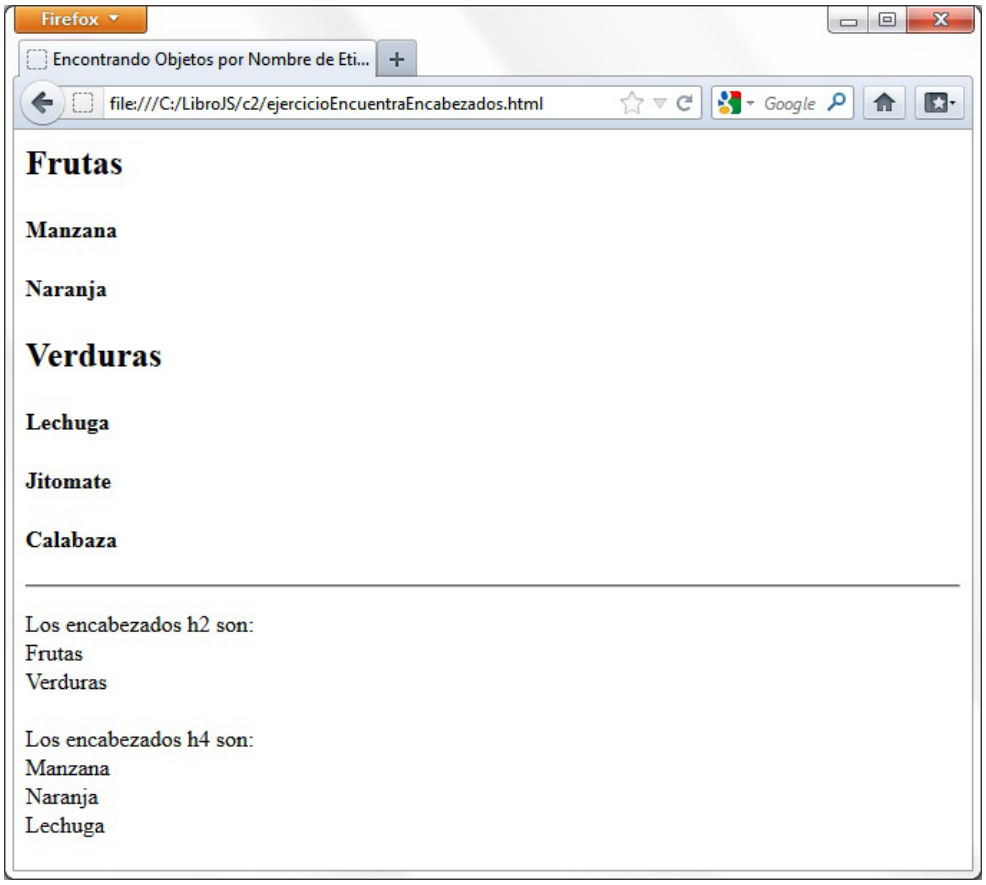
Figura 2.3. Encontrando párrafos por nombre de etiqueta



Ejercicio: Encontrando encabezados h2 y h4

Escribe un documento HTML con el nombre *ejercicioEncuentraEncabezados.html*. El documento debe visualizarse en el navegador web como se muestra en la figura 2.4. En la página web hay dos encabezados `<h2>`: `Frutas` y `Verduras`, más cinco encabezados `<h4>`: `Manzana`, `Naranja`, `Lechuga`, `Jitomate` y `Calabaza`. El contenido que se encuentra después de la línea horizontal debe ser generado a partir del contenido de los encabezados `<h2>` y `<h4>`. Para este ejercicio debes encontrar los encabezados en la página web para realizar lo que se solicita.

Figura 2.4. Página web para encontrar encabezados h2 y h4



Encontrando objetos por el nombre de su clase

Es posible encontrar elementos HTML con DOM utilizando el nombre de la clase a la que pertenecen en la página web. Para encontrar elementos por el nombre de su clase se utiliza el método `getElementsByClassName()` del objeto `document`, el cual recibe como parámetro el valor del atributo `class` de los elementos que se desea encontrar. El método `getElementsByClassName()` regresa un arreglo con los elementos HTML que tienen la clase especificada si existen o `null` si no los encuentra.

Ejemplo: Encontrando párrafos por el nombre de su clase

En este ejemplo se realizará un documento HTML para mostrar cómo encontrar elementos HTML por el nombre de su clase.

Con el contenido que se muestra en el listado 2.3, se creará un documento HTML con nombre *encuentraParrafosPorClase.html*; después, se abrirá en un navegador web.

Listado 2.3. Archivo *encuentraParrafosPorClase.html*

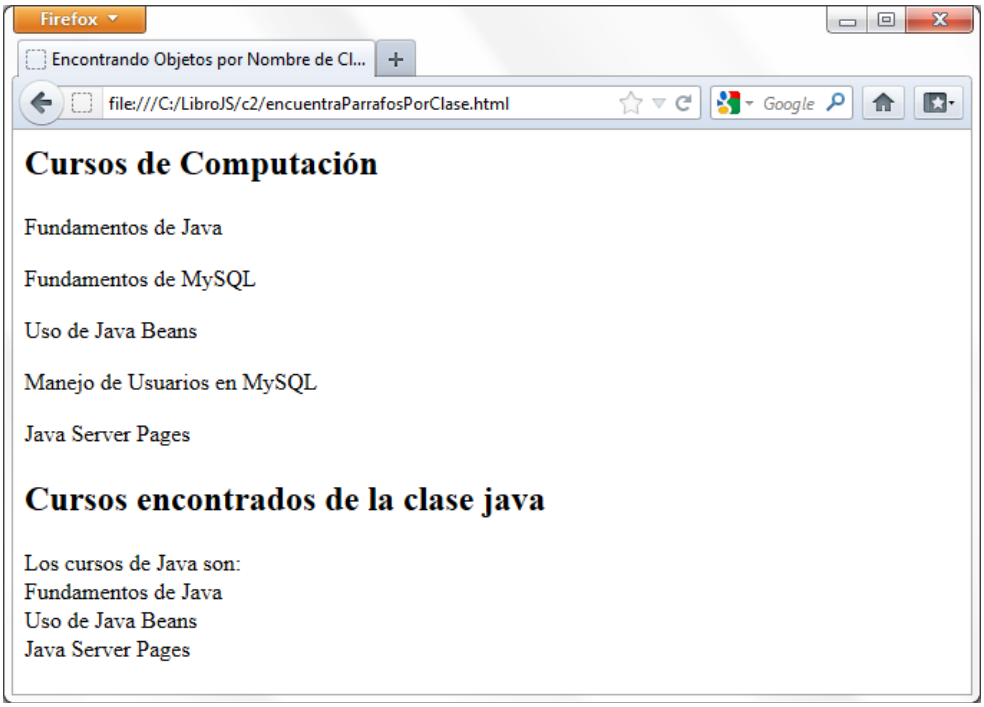
```
1   <html>
2       <head>
3           <title>Encontrando Objetos por Nombre de Clase</title>
4       <body>
5           <h2>Cursos de Computaci&oacute;n</h2>
6           <p class="java">Fundamentos de Java</p>
7           <p class="sql">Fundamentos de MySQL</p>
8           <p class="java">Uso de Java Beans</p>
9           <p class="sql">Manejo de Usuarios en MySQL</p>
10          <p class="java">Java Server Pages</p>
11          <h2>Cursos encontrados de la clase java</h2>
12          <p id="texto"></p>
13          <script type="text/javascript">
14              cursos = document.getElementsByClassName("java");
15              miTexto = document.getElementById("texto");
16              miTexto.innerHTML = "Los cursos de Java son:" +
17                                  "<br />" + cursos[0].innerHTML +
18                                  "<br />" + cursos[1].innerHTML +
19                                  "<br />" + cursos[2].innerHTML;
20          </script>
21      </body>
22  </html>
```

El listado 2.3 crea un documento HTML con seis párrafos y un segmento de código JavaScript. En la línea 6 del listado se encuentra el primer párrafo con el atributo `class="java"`, el cual determina la clase del párrafo, con el texto `Fundamentos de Java`. En la línea 7 está el segundo párrafo con el atributo `class="sql"`, con el texto `Fundamentos de MySQL`. El tercer párrafo se encuentra en la línea 8 con el atributo `class="java"`, con el texto `Uso de Java Beans`. El cuarto párrafo está en la línea 9 con el atributo `class="sql"`, con el texto `Manejo de Usuarios en MySQL`. En la línea 10 se encuentra el quinto párrafo con el atributo `class="java"`, con el texto `Java Server Pages`. En la línea 12 se encuentra el sexto párrafo con su

atributo `id="texto"`, el cual representa el identificador de ese elemento HTML y no tiene contenido entre su etiqueta inicial `<p>` y su etiqueta final `</p>`, ya que este párrafo será utilizado para escribir su contenido mediante JavaScript. Las líneas 13 a 20 contienen sentencias JavaScript delimitadas por la etiqueta inicial `<script>` y la etiqueta final `</script>`. La primera de las sentencias, en la línea 14, se encarga de encontrar todos los elementos HTML cuyo nombre de clase sea `java`, mediante la invocación al método `getElementsByClassName("java")` del objeto `document`, y almacena la referencia al arreglo de los elementos encontrados en la variable `cursos`. La línea 15 encuentra el elemento HTML cuyo identificador es `texto`, mediante la invocación al método `getElementById("texto")` del objeto `document`, y almacena la referencia al elemento en la variable `miTexto`. Finalmente, las líneas 16 a 19 concatenan la cadena de texto `"Los cursos de Java son:"` con el valor del atributo `innerHTML` de los tres elementos del arreglo de `cursos`, los cuales son accedidos a través de su índice: `cursos[0]`, `cursos[1]`, y `cursos[2]`, y corresponden a los párrafos 1, 3 y 5, cuyo nombre de clase es `java`. La cadena resultante es asignada al atributo `innerHTML` del objeto `miTexto`, el cual corresponde al sexto párrafo del documento HTML. Con estas sentencias JavaScript, el resultado de este ejemplo es que el sexto párrafo de la página web tendrá como contenido el texto `"Los cursos de Java son: <br /> Fundamentos de Java <br /> Uso de Java Beans <br /> Java Server Pages"`. Esto se realiza mediante la recuperación de elementos HTML con el método `getElementsByClassName()` y usando el atributo `innerHTML` para modificar el contenido de los elementos.

La figura 2.5 muestra la página web *encuentraParrafosPorClase.html* de este ejemplo visualizada en un navegador web, donde se observa que el contenido del sexto párrafo es el texto `"Los cursos de Java son: <br /> Fundamentos de Java <br /> Uso de Java Beans <br /> Java Server Pages"`, el cual fue generado dinámicamente, como ya se explicó.

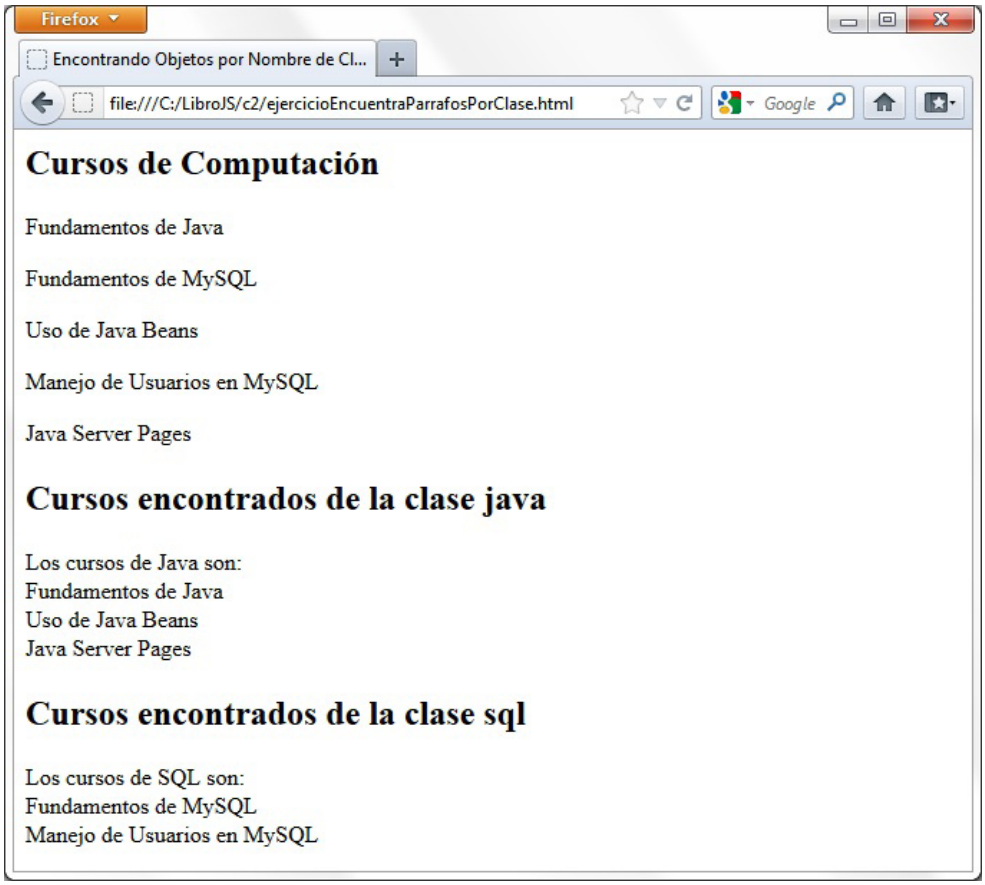
Figura 2.5. Encontrando párrafos por nombre de clase



Ejercicio: Encontrando párrafos con clase java y sql

Escribe un HTML con el nombre *ejercicioEncuentraParrafosPorClase.html*. Este documento debe visualizarse en el navegador web como se muestra en la figura 2.6. Para este ejercicio puedes tomar como base el ejemplo anterior. El contenido que se encuentra debajo de los encabezados *Cursos encontrados de la clase java* y *Cursos encontrados de la clase sql* debe ser generado a partir del contenido que se encuentra debajo del encabezado *Cursos de Computación*. Para este ejercicio debes encontrar los elementos HTML de acuerdo con su clase en la página web para realizar lo que se solicita.

Figura 2.6. Página web para encontrar párrafos con clase java y sql



Encontrando objetos por colecciones

También se pueden encontrar elementos HTML con DOM utilizando colecciones de objetos predefinidas en la página web. Para encontrar elementos por colecciones de objetos se utilizan las colecciones del objeto `document`, las cuales son representadas como arreglos. Algunas colecciones son:

`document.anchors` recupera una colección de todas las anclas de la página web.
`document.links` recupera una colección de todos los hipervínculos de la página web.

`document.forms` recupera una colección de todos los formularios de la página web.
`document.images` recupera una colección de todas las imágenes de la página web.

Para una lista completa de las colecciones de objetos que pueden ser utilizadas, puede consultarse la sección "Resumen de métodos y atributos" al final de este capítulo.

Ejemplo: Encontrando hipervínculos mediante su colección

En este ejemplo se realizará un documento HTML para mostrar cómo encontrar elementos HTML mediante una colección de objetos.

A partir del contenido que se muestra en el listado 2.4, se creará un documento HTML con el nombre *encuentraHipervinculosPorColeccion.html*; después, se abrirá en un navegador web.

Listado 2.4. Archivo *encuentraHipervinculosPorColeccion.html*

```
<html>
    <head>
        <title>Encontrando Objetos por Colecciones</title>
    <body>
        <h2>Hiperv&iacute;nculos a Sitios de la UAM</h2>
        <a href="http://www.cua.uam.mx">UAM Cuajimalpa</a>
        <br />
        <a href="http://www.azc.uam.mx">UAM Azcapotzalco</a>
        <br />
        <a href="http://www.izt.uam.mx">UAM Iztapalapa</a>
        <br />
        <p id="texto"></p>
        <script type="text/javascript">
            links = document.links;
            miTexto = document.getElementById("texto");
            miTexto.innerHTML = "En esta p&aacute;gina web hay" +
                                document.links.length +
                                "hiperv&iacute;nculos y son:" +
                                "<br />" + links[0].innerHTML +
                                "<br />" + links[1].innerHTML +
                                "<br />" + links[2].innerHTML;
        </script>
    </body>
</html>
```

El listado 2.4 crea un documento HTML con tres hipervínculos, un párrafo y un segmento de código JavaScript. En la líneas 6, 8 y 10 se encuentran los tres hipervínculos, cada uno con su atributo `href` y su respectivo texto delimitado por la etiqueta inicial `<a>` y la etiqueta final `</a>`. En la línea 12 se encuentra el párrafo con su atributo `id="texto"`, el cual representa el identificador de ese elemento HTML y no tiene contenido entre su etiqueta inicial `<p>` y su etiqueta final `</p>`, ya que este párrafo será utilizado para escribir su contenido mediante JavaScript. Las líneas 13 a 22 contienen sentencias JavaScript delimitadas por la etiqueta inicial `<script>` y la etiqueta final `</script>`. La primera de las sentencias, en la línea 14, se encarga de recuperar la colección de todos los hipervínculos que se encuentran en la página web por medio del método `document.links`, el cual regresa un arreglo cuya referencia es almacenada en la variable `links`. La línea 15 encuentra el elemento HTML cuyo identificador es `texto`, mediante la invocación al método `getElementById("texto")` del objeto `document`, y almacena la referencia al elemento en la variable `miTexto`. Finalmente, las líneas 16 a 21 concatenan una cadena de texto que contiene el número de hipervínculos que se encuentran en la página web, mediante el atributo `document.links.length`, y cada uno de los textos de los hipervínculos, los cuales son recuperados a través de su índice: `links[0]`, `links[1]`, y `links[2]`, y corresponden a los hipervínculos encontrados en la página web. La cadena resultante es asignada al atributo `innerHTML` del objeto `miTexto`, el cual corresponde al párrafo que se encuentra en la línea 12 del documento HTML. Con estas sentencias JavaScript, el resultado es que el párrafo de la página web tendrá como contenido el texto "`En esta p&aacute;gina web hay 3 hiperv&iacute;nculos y son: <br /> UAM Cuajimalpa <br /> UAM Azcapotzalco <br /> UAM Iztapalapa`". Esto se realiza mediante la recuperación de elementos HTML a través de la colección `document.links` y usando el atributo `innerHTML` para modificar el contenido de los elementos.

La figura 2.7 muestra la página web *encuentraHipervinculosPorColeccion.html* de este ejemplo, visualizada en un navegador web, donde se observa que el contenido del párrafo tiene el número de hipervínculos y el texto que se encuentra en cada uno de ellos, el cual fue generado dinámicamente, como ya se explicó.

Figura 2.7. Encontrando hipervínculos mediante su colección

Ejercicio: Encontrando hipervínculos con sus URL

Escribe un HTML con el nombre *ejercicioEncuentraHipervinculos.html*. Este documento debe visualizarse en el navegador web como se muestra en la figura 2.8. Para este ejercicio puedes tomar como base el ejemplo anterior. La tabla que se muestra debe ser generada a partir del contenido que se encuentra en los hipervínculos: la primera columna tiene el texto de los hipervínculos y la segunda el URL de los hipervínculos. Para este ejercicio debes encontrar los hipervínculos mediante la colección `document.links` de la página web para realizar lo que se solicita.

Figura 2.8. Página web para encontrar hipervínculos mediante su colección

Resumen de métodos y atributos

En la tabla 2.1 se presenta un resumen de los métodos y atributos revisados en este capítulo. También se incorporan algunos otros métodos y atributos que pueden ser utilizados de manera similar a los que se revisaron en el capítulo.

Tabla 2.1 Métodos y atributos del capítulo 2

Método o Atributo	Descripción
`document.getElementById()`	Encuentra un elemento por su identificador (atributo `id` del elemento)
`document.getElementsByTagName()`	Encuentra elementos por el nombre de su etiqueta
`document.getElementsByClassName()`	Encuentra elementos por el nombre de su clase (atributo `class` del elemento)
`document.anchors`	Colección de objetos ancla
`document.body`	Objeto con el cuerpo del documento
`document.forms`	Colección de objetos formulario
`document.head`	Objeto con la cabecera del documento
`document.images`	Colección de objetos imagen
`document.links`	Colección de objetos hipervínculo
`document.scripts`	Colección de objetos script
`document.title`	Objeto con el título del documento
`element.innerHTML`	Atributo con el contenido HTML del elemento específico

Capítulo 3. Manejo de eventos

En este capítulo se describe cómo manejar los eventos que ocurren en un documento HTML y responder a ellos mediante la ejecución de código JavaScript. El manejo de eventos permite tener una página web dinámica que responde con acciones a las interacciones que ocurren dentro del documento HTML, tal como cuando un usuario da click sobre algún elemento de la página web. En este capítulo también se proporcionan varios ejemplos y ejercicios para ilustrar el manejo de eventos sobre elementos HTML.

Los objetivos que deben cumplirse al final de este capítulo son:

- Conocer cómo se manejan y cuándo se disparan eventos en una página web.
- Escribir funciones y segmentos de código JavaScript para manejar eventos.
- Utilizar los atributos que representan los diferentes eventos en elementos HTML.
- Asociar los segmentos y funciones JavaScript con atributos que representan eventos.
- Identificar cuándo se disparan algunos eventos específicos, como *onclick, onmouseover, onmouseout*, entre otros.

Eventos en un documento HTML

A través de DOM es posible que JavaScript reaccione a los eventos que ocurren en un documento HTML, una vez que este está siendo cargado en un navegador web. Cuando un evento ocurre en un documento HTML se puede ejecutar un segmento de código JavaScript como respuesta a tal evento, como cuando un usuario da click sobre algún elemento HTML de una página web.

Para ejecutar un segmento de código JavaScript cuando un evento ocurre, es necesario añadir un atributo del evento correspondiente al elemento

HTML sobre el cual se dispara el evento. Algunos eventos son: cuando un usuario da click sobre algún elemento de la página web, cuando la página web ha terminado de cargarse en el navegador web, cuando una imagen ha sido cargada en la página web, cuando el mouse se mueve sobre un elemento de la página web, cuando el texto de un elemento de captura ha cambiado, entre otros. En las siguientes secciones se presentarán ejemplos y ejercicios que muestran la ejecución de código JavaScript en respuesta a diferentes eventos.

Ejemplo: Cambiando texto en respuesta a un click

En este ejemplo se realizará un documento HTML para mostrar cómo ejecutar código JavaScript en respuesta a un click sobre un elemento HTML.

A partir del contenido que se muestra en el listado 3.1, se creará un documento HTML con el nombre *eventoClickCambiaTexto.html*; después, se abrirá en un navegador web.

Listado 3.1. Archivo *eventoClickCambiaTexto.html*

```
1  <html>
2      <head>
3          <title>Cambiando Texto en Respuesta a Click</title>
4      </head>
5      <body>
6          <h1>Da click en el texto de abajo para cambiarlo</h1>
7          <h2 onclick="this.innerHTML += 'Texto cambiado!'">
8              Texto original.</h2>
9      </body>
10 </html>
```

El listado 3.1 crea un documento HTML con dos encabezados. En la línea 6 del listado se encuentra un encabezado `<h1>`, el cual da instrucciones para dar click sobre el texto que se encuentra en el siguiente encabezado `<h2>` para cambiar su contenido. En la línea 7 y 8 se encuentra el encabezado `<h2>` con el atributo `onclick="this.innerHTML += 'Texto cambiado!'"`, el cual indica que el código JavaScript que se encuentra como valor del atributo `onclick` será ejecutado cuando el usuario dé click sobre el texto del encabezado, esto es cuando ocurra el evento `onclick` sobre el elemento HTML. El atributo `onclick` se utiliza para referirse a la acción de dar click sobre el elemento HTML en el cual se coloca.

En este ejemplo, cada vez que el usuario dé click sobre el texto del encabezado `<h2>` se ejecutará el código JavaScript asociado al evento `onclick`, el cual concatenará el texto `Texto cambiado!` con el texto que se encuentra actualmente en el encabezado, determinado por el atributo `this.innerHTML`, donde la palabra `this` se refiere al elemento actual y el atributo `innerHTML` tiene el contenido de dicho elemento HTML.

La figura 3.1 muestra la página web *eventoClickCambiaTexto.html* de este ejemplo visualizada en un navegador web, sin haber dado click sobre el texto del encabezado `<h2>`.

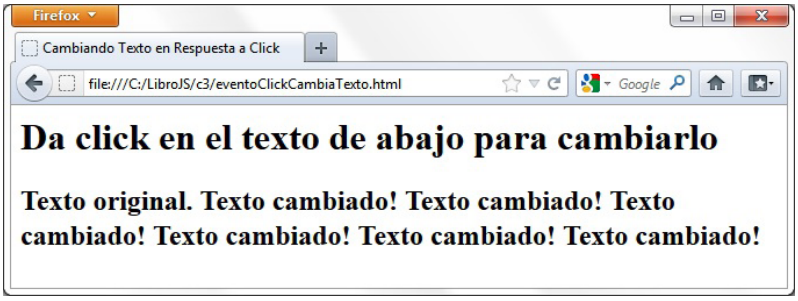
La figura 3.2 muestra la página web *eventoClickCambiaTexto.html* después de que el usuario ha dado click una vez sobre el texto del encabezado `<h2>`.

La figura 3.3 muestra la página web *eventoClickCambiaTexto.html* después de que el usuario ha dado click varias veces sobre el texto del encabezado <h2>. Se observa que cada vez que el usuario da click sobre el texto del encabezado se ejecuta el código JavaScript asociado con el evento `onclick` del encabezado `<h2>`.

Figura 3.1. Manejo del evento onclick sobre un encabezado

Figura 3.2. Página web después de haber dado click una vez sobre el encabezado

Figura 3.3. Página web después de haber dado click varias veces sobre el encabezado



Ejercicio: Cambiando texto de un párrafo en respuesta a clicks

Escribe un documento HTML con el nombre *ejercicioClickParrafos.html*. El documento debe visualizarse en el navegador web como se muestra en la figura 3.4, la cual tiene cuatro párrafos. Para este ejercicio puedes tomar como base el ejemplo anterior. El texto del párrafo, que se encuentra debajo de la línea horizontal, debe ser concatenado con el texto de los párrafos de arriba cada vez que el usuario dé click sobre alguno de ellos. Por ejemplo, en la figura 3.5 el usuario ha dado click en tres ocasiones sobre el `Párrafo 2`, por lo que el texto del cuarto párrafo cambió y se le concatenó tres veces el texto `Párrafo 2`. En la figura 3.6 se observa que el usuario ha dado click varias veces sobre los tres párrafos, por lo que el texto del cuarto párrafo cambió y se le han concatenado los textos `Párrafo 1`, `Párrafo 2` y `Párrafo 3` en varias ocasiones.

Figura 3.4. Manejo del evento onclick sobre tres párrafos

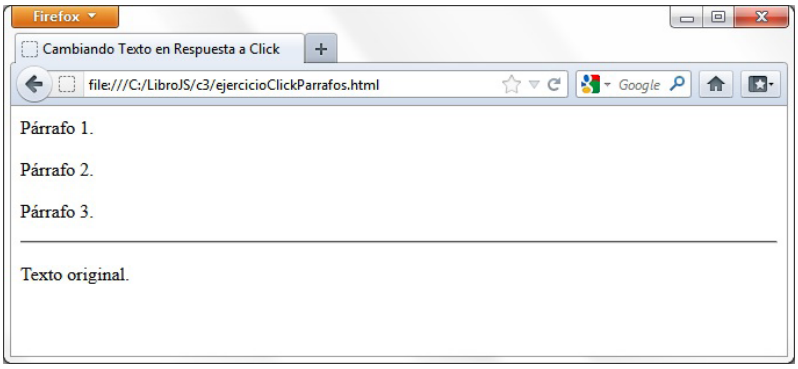


Figura 3.5. Página web después de haber dado click tres veces sobre Párrafo 2

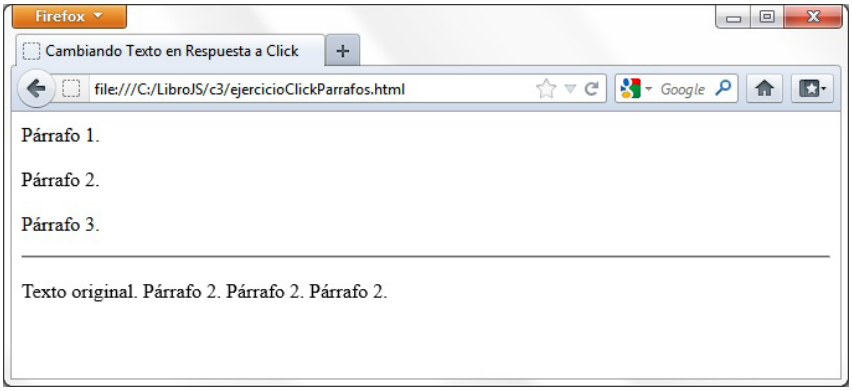
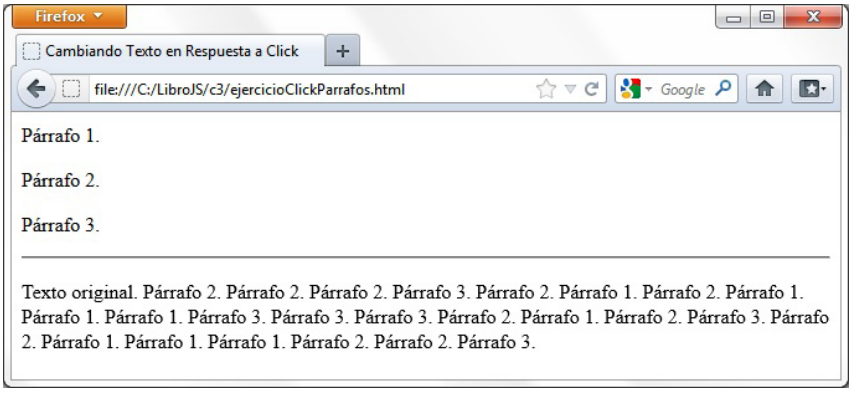


Figura 3.6. Página web después de haber dado click varias veces sobre los párrafos



Funciones JavaScript para manejar eventos

Como se observó en el ejemplo de la sección anterior, es posible ejecutar código JavaScript en respuesta a cualquier evento que ocurre en un documento HTML. Esto se llevó a cabo escribiendo directamente el código JavaScript como valor del atributo correspondiente al evento, dentro del elemento en el cual se deseaba manejar el evento, de la siguiente manera: `onclick = "this.innerHTML += 'Texto cambiado!'"`. Si bien es posible hacer esto para un código de una línea, o unas cuantas, sería poco legible si se hiciera para segmentos de código JavaScript más extensos. Para estos casos es preferible colocar ese segmento de código en una función JavaScript, la cual puede ser invocada para manejar cualquier evento; además, de esta manera se promueve la reutilización de código, ya que la misma función puede ser invocada por varios eventos si así se desea.

Las funciones JavaScript deben colocarse entre la etiqueta inicial <script> y la etiqueta final </script>, las cuales a su vez deben ir en la cabecera del documento HTML, entre la etiqueta inicial <head> y la etiqueta final</head>.

Ejemplo: Llamando a una función en respuesta a un click

En este ejemplo se realizará un documento HTML para mostrar cómo ejecutar una función JavaScript en respuesta a un click sobre un elemento HTML. La diferencia de este ejemplo, respecto al anterior, es que se invocará una función JavaScript que ejecutará el código, mientras que en el ejemplo anterior el código no estaba en una función, sino directamente como valor del atributo onclick del elemento HTML.

A partir del contenido que se muestra en el listado 3.2, se creará un documento HTML con el nombre *eventoClickCambiaTextoFuncion.html*; después, se abrirá en un navegador web.

Listado 3.2. Archivo *eventoClickCambiaTextoFuncion.html*

```
1   <html>
2       <head>
3          <title>Llamando Funci&oacute;n en Respuesta a Click</title>
4           <script type="text/javascript">
5               function cambiaTexto(obj){
6                   obj.innerHTML += "Función JavaScript ejecutada!";
7               }
8           </script>
9       </head>
10      <body>
11          <h1>Da click en el texto de abajo para cambiarlo</h1>
12          <h2 onclick="cambiaTexto(this);">
13              Texto original.</h2>
14      </body>
15  </html>
```

El listado 3.2 crea un documento HTML con dos encabezados y un segmento de código JavaScript con una función. En la línea 11 del listado se encuentra un encabezado <h1> que da instrucciones para dar click sobre el texto que se encuentra en el siguiente encabezado <h2> con el fin de cambiar su contenido. En las líneas 12 y 13 se encuentra el encabezado <h2> con el atributo onclick="cambiaTexto(this);", el cual indica que la

función JavaScript que se encuentra como valor del atributo `onclick` será ejecutada cuando el usuario dé click sobre el texto del encabezado, esto es cuando ocurra el evento `onclick` sobre el elemento HTML. La función JavaScript `cambiaTexto(obj)` que es invocada cuando ocurre el evento `onclick` está definida en el segmento de código JavaScript, en las líneas 4 a 8, el cual está delimitado por la etiqueta inicial `<script>` y la etiqueta final `</script>`. La función `cambiaTexto(obj)` recibe un parámetro que representa el elemento HTML sobre el cual se modificará el texto y ejecuta, en su línea 6, una concatenación del texto `Función JavaScript ejecutada!` con el texto que se encuentra actualmente en el elemento, que es pasado como parámetro y determinado por `obj.innerHTML`, donde la palabra `obj` se refiere al elemento pasado como parámetro y el atributo `innerHTML` tiene el contenido de dicho elemento HTML.

La figura 3.7 muestra la página web *eventoClickCambiaTextoFuncion.html* de este ejemplo visualizada en un navegador web, sin haber dado click sobre el texto del encabezado `<h2>`.

La figura 3.8 muestra la página web *eventoClickCambiaTextoFuncion.html* después de que el usuario ha dado click una vez sobre el texto del encabezado `<h2>`.

La figura 3.9 muestra la página web *eventoClickCambiaTextoFuncion.html* después de que el usuario ha dado click varias veces sobre el texto del encabezado <h2>. Se observa que cada vez que el usuario da click sobre el texto del encabezado, se ejecuta la función JavaScript asociada con el evento `onclick` del encabezado `<h2>`.

Figura 3.7. Manejo del evento onclick sobre un encabezado

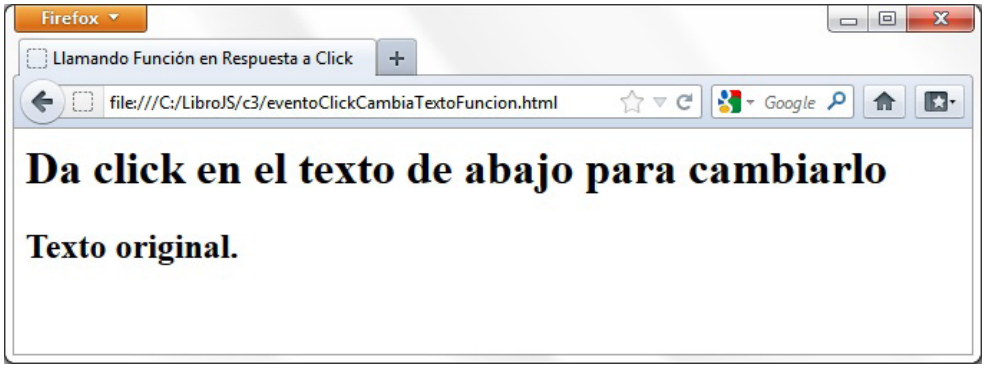


Figura 3.8. Página web después de haber dado click una vez sobre el encabezado

Figura 3.9. Página web después de haber dado click varias veces sobre el encabezado

Ejercicio: Llamando a una función para cambiar texto de un párrafo

Escribe un documento HTML con el nombre *ejercicioClickParrafosFuncion.html*. El documento debe visualizarse en el navegador web como se muestra en la figura 3.10, en la cual se observan cuatro párrafos. Este ejercicio es el mismo que se presentó en la sección anterior pero en esta ocasión se solicita realizarlo haciendo uso de una función JavaScript para colocar el código que sea requerido. El texto del párrafo que se encuentra debajo de la línea horizontal debe ser concatenado con el texto de los párrafos de arriba cada vez que el usuario dé click sobre alguno de ellos. Por ejemplo, en la figura 3.11 el usuario ha dado click en dos ocasiones sobre el `Párrafo 2` y en dos ocasiones sobre el `Párrafo 3`, por lo que el texto del cuarto párrafo cambió y se le concatenó dos veces el texto `Párrafo 2`, y dos veces el texto `Párrafo 3`. En la figura 3.12 se observa que el usuario ha dado click varias veces sobre los tres párrafos, por ello el texto del cuarto párrafo cambió y se le han concatenado los textos `Párrafo 1`, `Párrafo 2` y `Párrafo 3` en varias ocasiones.

Figura 3.10. Manejo del evento onclick sobre tres párrafos con una función

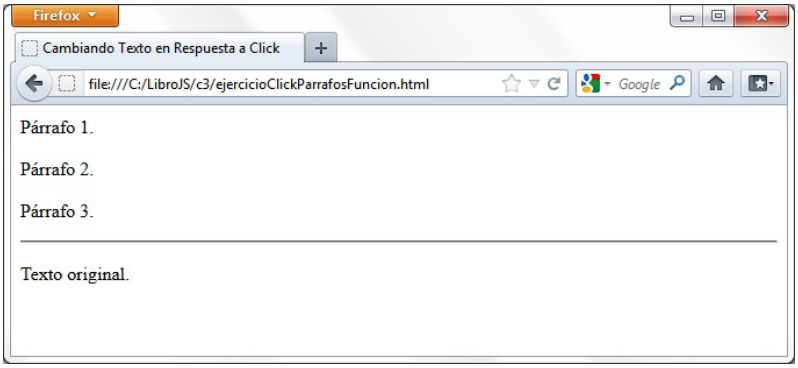


Figura 3.11. Página web después de haber dado click dos veces sobre Párrafo 2 y Párrafo 3

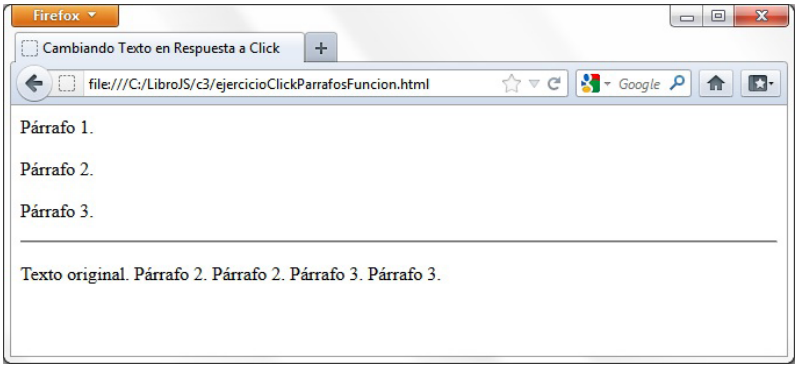
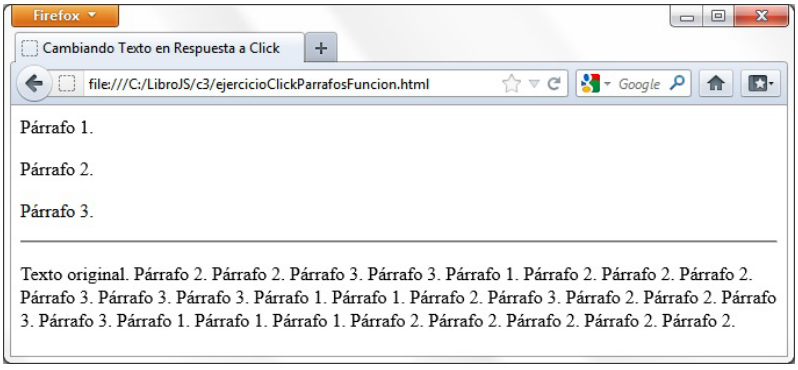


Figura 3.12. Página web después de haber dado click varias veces sobre los párrafos



Eventos onmouseover y onmouseout

Los eventos `onmouseover` y `onmouseout` también pueden ser manejados a través de la ejecución de funciones JavaScript. El evento `onmouseover` se dispara cuando el mouse se mueve sobre un elemento HTML, mientras que el evento `onmouseout` se dispara cuando el mouse se mueve fuera del elemento HTML.

Para estos dos eventos también es necesario añadirlos como atributos del elemento HTML sobre el cual se desea manejarlos, como se ha hecho en las secciones anteriores. El valor del atributo debe contener la invocación a la función JavaScript que será ejecutada cuando el evento ocurra. Los atributos asociados a estos eventos son precisamente los atributos `onmouseover` y `onmouseout`.

Ejemplo: Cambiando texto en respuesta a onmouseover y onmouseout

En este ejemplo se realizará un documento HTML para mostrar cómo ejecutar funciones JavaScript en respuesta a los eventos `onmouseover` y `onmouseout` sobre un elemento HTML.

A partir del contenido que se muestra en el listado 3.3, se creará un documento HTML con el nombre *eventoOnMouseOverCambiaTexto.html*; después, se abrirá en un navegador web.

Listado 3.3. Archivo *eventoOnMouseOverCambiaTexto.html*

```
1  <html>
2      <head>
3          <title>Eventos OnMouseOver y OnMouseOut</title>
4          <script type="text/javascript">
5              function mouseSobre(obj){
6                  obj.innerHTML = 'Mouse SOBRE el elemento!';
7              }
8              function mouseFuera(obj){
9                  obj.innerHTML = 'Mouse FUERA del elemento!';
10             }
11         </script>
12     </head>
13     <body>
14         <h1 onmouseover="mouseSobre(this);"
15             onmouseout="mouseFuera(this);">
16             Mouse FUERA del elemento!</h1>
17     </body>
18 </html>
```

El listado 3.3 crea un documento HTML con un encabezado `<h1>` y un segmento de código JavaScript con dos funciones. En las líneas 14 a 16 del listado se encuentra un encabezado `<h1>` con dos atributos: el atributo `onmouseover="mouseSobre(this);"`, el cual indica que se ejecutará la función JavaScript `mouseSobre(this)` cuando se dispare el evento `onmouseover` sobre el elemento HTML, pasando como parámetro el elemento HTML actual, y el atributo `onmouseout="mouseFuera(this);"`, el cual indica que se ejecutará la función JavaScript `mouseFuera(this)` cuando se dispare el evento `onmouseout` sobre el elemento HTML, pasando como parámetro el elemento HTML actual. Las funciones JavaScript `mouseSobre(obj)` y `mouseFuera(obj)` están definidas en el segmento de código delimitado por la etiqueta inicial `<script>` y la etiqueta final `</script>`, en las líneas 4 a 11. La función `mouseSobre(obj)`, definida en las líneas 5 a 7, recibe como parámetro un objeto y cambia el contenido HTML de ese objeto a través de su atributo `innerHTML`, colocando el texto `Mouse SOBRE el elemento!`. La función `mouseFuera(obj)`, definida en las líneas 8 a 10, recibe como parámetro un objeto y cambia el contenido HTML de ese objeto a través de su atributo `innerHTML`, colocando el texto `Mouse FUERA del elemento!`.

En este ejemplo, cuando el usuario mueve el mouse encima del texto del encabezado `<h2>`, se disparará el evento `onmouseover` del encabezado, por lo que se ejecutará la función JavaScript asociada a ese evento, la cual cambia el texto del encabezado a `Mouse SOBRE el elemento!`. Cuando el usuario mueve el mouse fuera del texto del encabezado `<h2>`, se dispara el evento `onmouseout` del encabezado, por lo que se ejecutará la función JavaScript asociada a ese evento, la cual cambia el texto del encabezado a `Mouse FUERA del elemento!`.

La figura 3.13 muestra la página web *eventoOnMouseOverCambiaTexto.html* de este ejemplo visualizada en un navegador web.

La figura 3.14 muestra la página web *eventoOnMouseOverCambiaTexto.html* cuando el usuario mueve el mouse encima del texto del encabezado.

La figura 3.15 muestra la página web *eventoOnMouseOverCambiaTexto.html* después de que el usuario ha movido el mouse fuera del texto del encabezado.

Figura 3.13. Manejo de eventos onmouseover y onmouseout sobre un encabezado

Figura 3.14. Página web después de haber movido el mouse sobre el encabezado

Figura 3.15. Página web después de haber movido el mouse fuera del encabezado

Ejercicio: Mostrando texto en respuesta a onmouseover y onmouseout

Escribe un HTML con el nombre *ejercicioOnMouseOverCambiaTexto.html*. Este documento debe visualizarse en el navegador web como se muestra en la figura 3.16, en la cual se observan dos encabezados. Cuando el usuario mueve el mouse encima del encabezado `HTML`, el texto `Definici&oacute;n: Hypertext Markup Language` debe aparecer, como se observa en la figura 3.17. Cuando se mueve fuera del encabezado `HTML`, el texto debe desaparecer, como se muestra originalmente en la figura 3.16. Cuando el usuario mueve el mouse encima del encabezado `CSS`, el texto `Definici&oacute;n: Cascading Style Sheet` debe aparecer, como se observa en la figura 3.18. Cuando se mueve fuera del encabezado `CSS`, el texto debe desaparecer, como se muestra originalmente en la figura 3.16.

Figura 3.16. Manejo de eventos onmouseover y onmouseout sobre encabezados

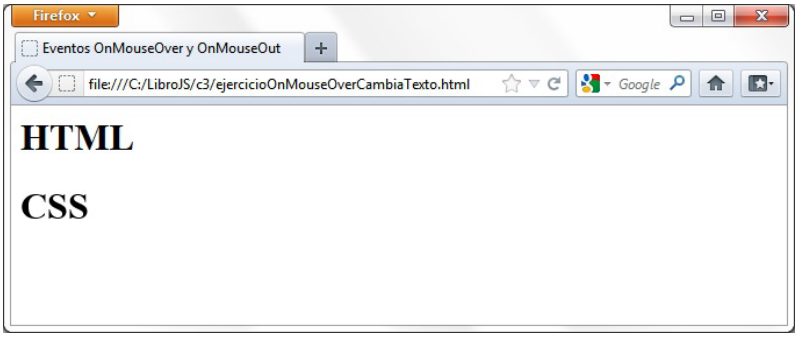


Figura 3.17. Página web después de haber movido el mouse sobre el encabezado HTML

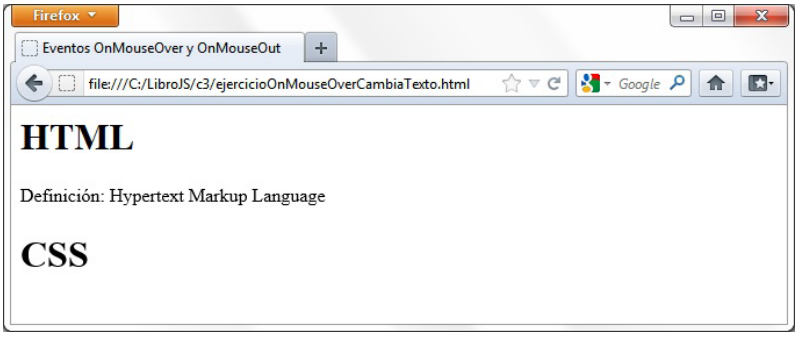
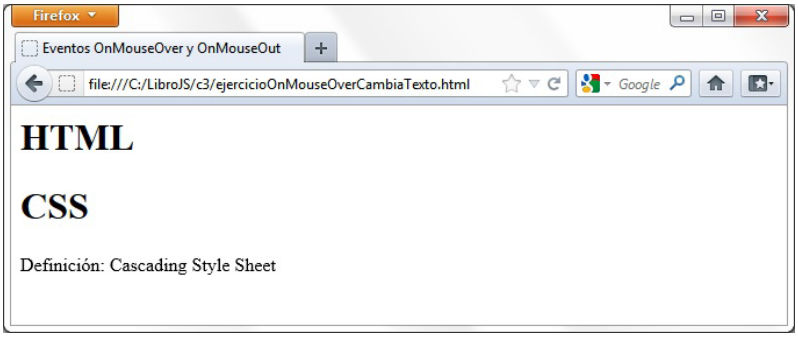


Figura 3.18. Página web después de haber movido el mouse sobre el encabezado CSS



Resumen de eventos

En la tabla 3.1 se presenta un resumen de los eventos revisados en este capítulo. También se incorporan otros eventos importantes que pueden ser utilizados de manera similar.

Tabla 3.1 Eventos y sus descripciones

Evento	Descripción
`onclick`	Se dispara cuando el usuario da click sobre un elemento
`ondblclick`	Se dispara cuando el usuario da doble click sobre un elemento
`onmousedown`	Se dispara cuando el usuario presiona el botón del mouse sobre un elemento
`onmouseup`	Se dispara cuando el usuario libera (deja de presionar) el botón del mouse sobre un elemento
`onmouseover`	Se dispara cuando el usuario mueve el mouse sobre un elemento
`onmouseout`	Se dispara cuando el usuario mueve el mouse fuera de un elemento
`onkeydown`	Se dispara cuando el usuario está presionando una tecla
`onkeypress`	Se dispara cuando el usuario presiona una tecla
`onkeyup`	Se dispara cuando el usuario libera (deja de presionar) una tecla
`onload`	Se dispara cuando un objeto se ha cargado
`onpageshow`	Se dispara cuando el usuario navega hacia una página web
`onpagehide`	Se dispara cuando el usuario deja una página web
`onblur`	Se dispara cuando un elemento pierde el foco
`onchange`	Se dispara cuando el contenido de un elemento de captura, la selección o el estado han cambiado. Esto aplica para los elementos `<input>`, `<select>` y `<textarea>`
`onfocus`	Se dispara cuando un elemento obtiene el foco
`oninput`	Se dispara cuando un elemento recibe entrada del usuario
`onreset`	Se dispara cuando un formulario es limpiado
`onselect`	Se dispara después de que el usuario ha seleccionado un segmento de texto. Esto aplica para los elementos `<input>` y `<textarea>`
`onsubmit`	Se dispara cuando se envía un formulario
`ondrag`	Se dispara cuando un elemento está siendo arrastrado
`ondragend`	Se dispara cuando el usuario ha terminado de arrastrar un elemento

Evento	Descripción
`ondragenter`	Se dispara cuando el elemento arrastrado ingresa a la zona donde se soltará
`ondragleave`	Se dispara cuando el elemento arrastrado sale de la zona donde se soltará
`ondragover`	Se dispara cuando el elemento arrastrado está sobre la zona donde se soltará
`ondragstart`	Se dispara cuando el usuario comienza a arrastrar un elemento
`ondrop`	Se dispara cuando el elemento arrastrado es soltado en la zona donde se soltará

Capítulo 4. Creación y eliminación de elementos HTML

En este capítulo se describe cómo crear y eliminar elementos en un documento HTML a través de DOM con JavaScript. Los elementos son los nodos del árbol que representa a un documento HTML, como párrafos `<p>`, encabezados `<h1>` a `<h6>`, imágenes `<img>`, tablas `<table>`, contenedores `<div>`, etc. La creación de un nuevo elemento incluye añadirlo a un elemento existente del documento HTML, mientras que la eliminación de un elemento involucra conocer el padre del elemento que se desea eliminar. En este capítulo también se proporcionan varios ejemplos y ejercicios para ilustrar cómo crear nuevos elementos y cómo eliminar elementos existentes en una página web.

Los objetivos que deben cumplirse al final de este capítulo son:

- Conocer cómo se crean nuevos elementos en una página web.
- Utilizar los métodos y atributos correspondientes para crear y añadir elementos a un documento HTML.
- Escribir funciones JavaScript para crear y añadir elementos a un documento HTML.
- Conocer cómo se eliminan elementos de una página web.
- Utilizar los métodos y atributos correspondientes para eliminar elementos existentes de un documento HTML.
- Escribir funciones JavaScript para eliminar elementos de un documento HTML.
- Asociar funciones JavaScript con eventos para crear y eliminar elementos de un documento HTML.

Creación de elementos HTML

Dentro de un documento HTML es posible añadir nuevos elementos a través de DOM con JavaScript, lo cual permite la generación dinámica de contenido dentro de una página web. Cada uno de los elementos son también conocidos como nodos dentro del árbol que representa a un documento HTML.

Para añadir un nuevo elemento o nodo a un documento HTML, primero se debe crear el elemento y después añadirlo a un elemento existente. Para crear un nuevo elemento se utiliza el método `createElement()` del objeto `document`. Para crear un nodo de texto se utiliza el método `createTextNode()` del objeto `document` y para añadir el nuevo elemento a otro elemento existente se utiliza el método `appendChild()`, el cual es un método que se debe ejecutar sobre el elemento existente.

Ejemplo: Creando un nuevo párrafo dentro de un contenedor

En este ejemplo se realizará un documento HTML para mostrar cómo crear un nuevo párrafo y añadirlo a un contenedor existente. El texto que tendrá el párrafo será definido mediante un campo de captura de texto y, en respuesta al evento `onclick` de un botón, se disparará la ejecución de una función JavaScript que creará el párrafo con el texto especificado y lo añadirá a un contenedor de la página web.

A partir del contenido que se muestra en el listado 4.1, se creará un documento HTML con el nombre *creaParrafo.html*; después, se abrirá en un navegador web.

Listado 4.1. Archivo *creaParrafo.html*

```
1   <html>
2       <head>
3           <title>Creaci&oacute;n de Elementos HTML</title>
4           <script type="text/javascript">
5               function creaParrafo(div, caja){
6                  panel = document.getElementById(div);
7                  cajaTexto = document.getElementById(caja);
8                  textoParrafo = cajaTexto.value;
9                  nuevoParrafo = document.createElement("p");
10                 nodoTexto = document.createTextNode(textoParrafo);
11                 nuevoParrafo.appendChild(nodoTexto);
12                 panel.appendChild(nuevoParrafo);
13              }
14          </script>
15      </head>
16      <body>
17          <h2>Creaci&oacute;n de Elementos HTML</h2>
18          <p>Escribe el texto para el nuevo p&aacute;rrafo
19             y da click el bot&oacute;n para crearlo:</p>
20          <input type="text" id="cajaTexto" />
21          <input type="button" value="Crea P&aacute;rrafo"
22                 onclick="creaParrafo('panel', 'cajaTexto');" />
23          <br /><br />
24          <div id="panel">
25              <p>Este es el p&aacute;rrafo 1.</p>
26              <p>Este es el p&aacute;rrafo 2.</p>
27          </div>
28      </body>
29  </html>
```

El listado 4.1 crea un documento HTML con un campo de captura de texto, un botón y un `<div>` que contiene dos párrafos. En la línea 20 del listado se encuentra un campo de captura de texto con su atributo `id="cajaTexto"`, el cual será empleado para capturar el texto que se desea que aparezca en el párrafo que será creado dinámicamente. En las líneas 21 y 22 se tiene un botón con el atributo `onclick="creaParrafo('panel', 'cajaTexto');"` indicando que, cuando ocurra el evento `onclick` del botón, la función JavaScript `creaParrafo()` será invocada, la cual toma dos parámetros: `panel` es el identificador del contenedor sobre el cual se generará el nuevo párrafo y `cajaTexto` es el identificador del campo de captura de texto del cual se obtendrá el texto para el nuevo párrafo. En las líneas 24 a 27 se define un `<div>`, el cual es un contenedor que agrupa los dos párrafos de las líneas 25 y 26, en este contenedor se agregará el nuevo párrafo. La función

JavaScript `creaParrafo()` que será ejecutada en respuesta al evento `onclick` del botón se encuentra definida de la línea 5 a la 13, en el segmento de código JavaScript, entre la etiqueta inicial `<script>` y la etiqueta final `</script>`. La función `creaParrafo()` toma dos parámetros: el primero es `div`, identificador del contenedor donde el nuevo párrafo será creado, y el segundo es `caja`, identificador del campo de captura de texto del cual se obtendrá el texto para el nuevo párrafo. La primera línea de la función, en la línea 6 del listado, encuentra al elemento que es el contenedor y almacena una referencia en la variable `panel`. La línea 7 encuentra al elemento que es el campo de captura de texto y almacena una referencia en la variable `cajaTexto`. La línea 8 obtiene el valor capturado en el campo de captura de texto y lo almacena en la variable `textoParrafo`. La creación del nuevo párrafo se lleva a cabo en la línea 9 con el método `createElement("p")` del objeto `document`, donde se indica que debe crearse un elemento "p", el cual corresponde a un párrafo, y su referencia se almacena en la variable `nuevoParrafo`. El texto del párrafo es creado como un nodo de texto en la línea 10 a través del método `createTextNode(textoParrafo)` del objeto `document`, el cual recibe como parámetro el texto que fue capturado en el campo de captura de texto y su referencia se almacena en la variable `nodoTexto`. La línea 11 se encarga de añadir el nodo de texto al párrafo mediante el método `appendChild(nodoTexto)` del párrafo `nuevoParrafo` que fue previamente creado. Finalmente, en la línea 12, el párrafo `nuevoParrafo` se añade al `panel` a través de su método `appendChild(nuevoParrafo)`. Con esta última línea el contenedor `panel` tendrá un nuevo párrafo añadido al final.

En este ejemplo, cada vez que el usuario dé click sobre el botón, se invocará la función JavaScript asociada con el evento `onclick`, la cual creará un nuevo párrafo con el texto que el usuario introduzca en el campo de captura de texto y lo añadirá al contenedor.

La figura 4.1 muestra la página web *creaParrafo.html* de este ejemplo visualizada en un navegador web, donde se observa el campo de captura de texto, el botón y los dos párrafos.

La figura 4.2 muestra nuevamente la página web *creaParrafo.html* después de que el usuario ha introducido texto en el campo de captura de texto y ha dado click al botón, con lo cual se ha creado un nuevo párrafo en el contenedor con el texto introducido en el campo de captura de texto.

La figura 4.3 muestra nuevamente la página web *creaParrafo.html* después de que el usuario ha introducido otro texto en el campo de captura de texto y ha dado click al botón, con lo cual se ha creado otro párrafo en el contenedor.

Figura 4.1. Página web *creaParrafo.html*

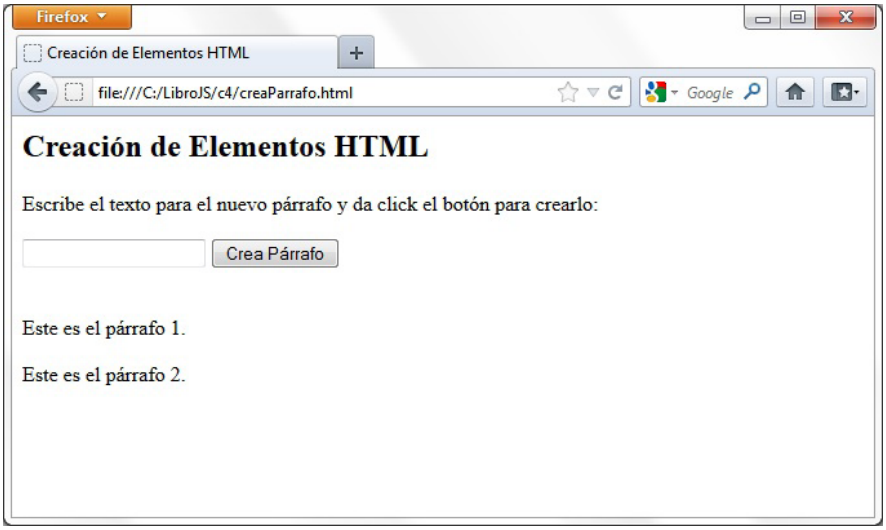


Figura 4.2. Página web después de haber introducido texto y dado click en el botón

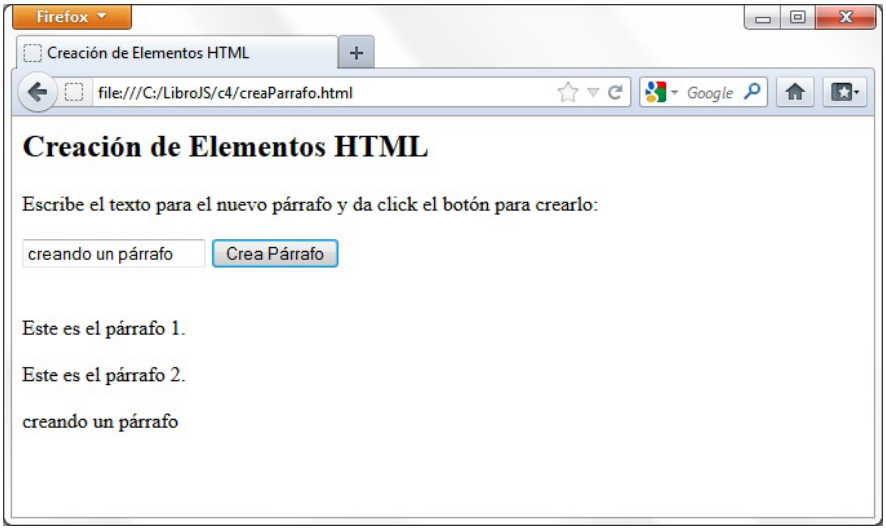
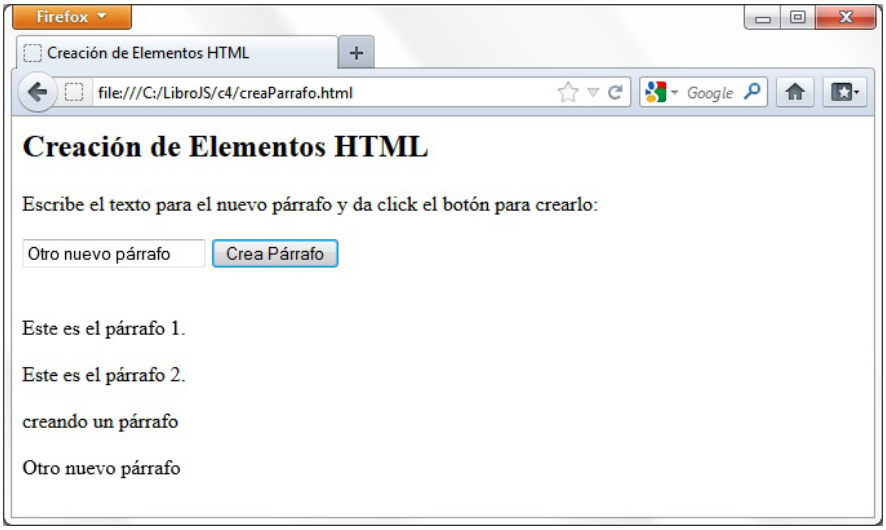


Figura 4.3. Página web después de haber introducido otro texto y dado click en el botón



Ejercicio: Creando un párrafo un número de veces determinado

Escribe un documento HTML con el nombre *ejercicioCreaParrafos.html*. El documento debe visualizarse en el navegador web como se muestra en la figura 4.4, en la cual se observan dos campos de captura de texto y un botón. Para este ejercicio puedes tomar como base el ejemplo anterior. En el primer campo de captura de texto el usuario introducirá un número, el cual indicará el número de párrafos que serán creados con el texto que se encuentra en el segundo campo de captura de texto. Por ejemplo, en la figura 4.5 el usuario indicó que debían generarse tres párrafos con el texto `Hola Mundo` que introdujo en el segundo campo de captura de texto, por lo que después de dar click en el botón, los tres párrafos con el texto `Hola Mundo` fueron generados.

Figura 4.4. Página web *ejercicioCreaParrafos.html*

Figura 4.5. Página web después de haber creado tres párrafos

Ejemplo: Creando campos de captura de texto en un contenedor

En este ejemplo se realizará un documento HTML para mostrar cómo crear varios campos de captura de texto y añadirlos a un contenedor existente. El número de campos de captura de texto que se creará será definido por el usuario mediante un campo de captura de texto y, en respuesta al evento `onclick` de un botón, se disparará la ejecución de una función JavaScript que creará el número de campos de captura de texto especificados y los añadirá a un contenedor de la página web.

Con el contenido que se muestra en el listado 4.2, se creará un documento HTML con el nombre *creaCamposTexto.html*; después, se abrirá en un navegador web.

Listado 4.2. Archivo *creaCamposTexto.html*

```
<html>
    <head>
        <title>Generador de Campos de Texto</title>
        <script type="text/javascript">
            function generaCajas(div, num){
                panel = document.getElementById(div);
                cajaTexto = document.getElementById(num);
                numCajas = cajaTexto.value;
                for (i=0; i<numCajas; i++){
                   nuevaCaja = document.createElement("input");
                   nuevaCaja.setAttribute("type", "text");
                   br = document.createElement("br");
                   panel.appendChild(nuevaCaja);
                   panel.appendChild(br);
                }
            }
        </script>
    </head>
    <body>
        <h2>Generador de Campos de Texto</h2>
        <p>Escribe el n&uacute;mero de campos de texto que
           deseas generar din&aacute;micamente:</p>
        <input type="text" id="cajaTexto" />
        <input type="button" value="Generar Cajas"
               onclick="generaCajas('panel', 'cajaTexto');" />
        <br /><br />
        <div id="panel"></div>
    </body>
</html>
```

El listado 4.2 crea un documento HTML con un campo de captura de texto, un botón y un vacío. En la línea 23 del listado se encuentra un campo de captura de texto con su atributo id="cajaTexto", el cual será empleado para capturar el número de campos de captura de texto que serán creados dinámicamente. En las líneas 24 y 25 se tiene un botón con el atributo onclick="generaCajas('panel', 'cajaTexto');" indicando que cuando ocurra el evento onclick del botón, la función JavaScript generaCajas() será invocada, la cual toma dos parámetros: panel es el identificador del contenedor sobre el cual se generarán los nuevos campos de captura de texto y cajaTexto es el identificador del campo de captura de texto del cual se obtendrá el número de campos de captura de texto que serán creados. En la línea 27 se define un con su atributo id="panel", el cual es un contenedor que está vacío, en este contenedor se agregarán los nuevos campos de captura de texto. La función JavaScript generaCajas() que será ejecutada en respuesta al evento onclick del botón se encuentra definida de la línea 5 a la 16, en el segmento de código JavaScript, entre la etiqueta inicial <script> y la etiqueta final </script>. La función generaCajas() toma dos parámetros: el primero es div, identificador del contenedor donde los nuevos campos de captura de texto serán creados y el segundo es num, identificador del campo de captura de texto del que se obtendrá el número de campos de captura de texto que serán creados. La primera línea de la función, en la línea 6 del listado, encuentra al elemento que es el contenedor y almacena una referencia en la variable panel; la línea 7 encuentra al elemento que es el campo de captura de texto y almacena una referencia en la variable cajaTexto; la línea 8 obtiene el valor capturado en el campo de captura de texto y lo almacena en la variable numCajas. La creación de los nuevos campos de captura de texto se lleva a cabo de la línea 9 a la 15, donde se tiene una estructura de control de flujo de programa for, también conocida como ciclo for, el cual se ejecuta el número de veces que el usuario introdujo en el campo de captura de texto. Dentro del ciclo for se ejecutan las sentencias que se encargan de crear el número de campos de captura de texto indicados por el usuario. En la línea 10 se crea un campo de captura de texto mediante el método createElement("input") del objeto document y su referencia se almacena en la variable nuevaCaja. Debido a que se requiere crear un campo de captura de texto, es necesario incluir el atributo correspondiente para indicarlo, por lo que en la línea 11 se crea el atributo type="text" mediante el método setAttribute("type", "text"), el cual se ejecuta sobre el elemento nuevaCaja recientemente creado. Para introducir un salto de línea entre cada uno de los campos de

captura de texto que se generarán dinámicamente, es necesario crear un elemento mediante el método `createElement("br")` del objeto `document` y almacenar su referencia en la variable `br`, como se muestra en la línea 12. Finalmente, en las líneas 13 y 14 se ejecuta el método `appendChild()` del `panel` para añadir el nuevo campo de captura de texto y el nuevo salto de línea, respectivamente. Este proceso de crear un nuevo campo de captura de texto y un salto de línea, y añadirlos al contenedor, se repite tantas veces como el usuario lo haya indicado.

En este ejemplo, cada vez que el usuario dé click sobre el botón, se invocará la función JavaScript asociada con el evento `onclick`, la cual creará el número de campos de captura de texto que el usuario introduzca.

La figura 4.6 muestra la página web *creaCamposTexto.html* de este ejemplo visualizada en un navegador web, donde se observa el campo de captura de texto y el botón.

La figura 4.7 muestra nuevamente la página web *creaCamposTexto.html* después de que el usuario ha introducido el número 2 en el campo de captura de texto y ha dado click al botón, con lo cual se han creado dos nuevos campos de captura de texto en el contenedor.

La figura 4.8 muestra nuevamente la página web *creaCamposTexto.html* después de que el usuario ha introducido el número 3 en el campo de captura de texto y ha dado click al botón, con lo cual se han creado otros tres nuevos campos de captura de texto en el contenedor para tener en total cinco.

Figura 4.6. Página web *creaCamposTexto.html*

Figura 4.7. Página web después de haber introducido un número y dado click en el botón

Figura 4.8. Página web después de haber introducido otro número y dado click en el botón

Ejercicio: Creando campos de captura de texto, botones y alertas

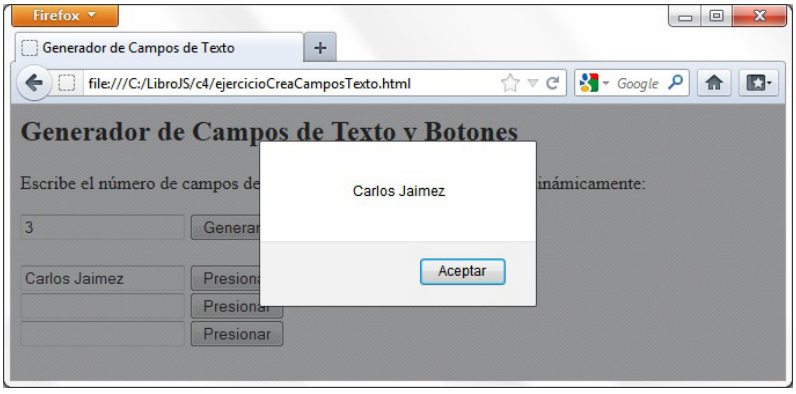
Escribe un documento HTML con el nombre *ejercicioCreaCamposTexto.html*. El documento debe visualizarse en el navegador web como se muestra en la figura 4.9, en la cual se observa un campo de captura de texto y un botón. Para este ejercicio puedes tomar como base el ejemplo anterior. En el campo de

captura de texto el usuario introducirá un número, el cual indicará el número de campos de captura de texto y botones que serán creados. Por ejemplo, en la figura 4.10 el usuario indicó que debían generarse tres campos de captura de texto con botones, por lo que después de dar click en el botón, los tres campos de captura de texto y botones fueron generados. Finalmente, en la figura 4.11, se observa que al presionar el primer botón se muestra una ventana de alerta con el texto que se encuentra en el primer campo de captura de texto; esto debe de funcionar para cada par de los campos de captura de texto y botones que se generen dinámicamente. Para mostrar una ventana de alerta se utiliza la función JavaScript `alert('Hola Mundo')`, donde la cadena `Hola Mundo` es el texto que debe mostrarse en la ventana de alerta.

Figura 4.9. Página web *ejercicioCreaCamposTexto.html*

Figura 4.10. Página web después de haber creado tres campos de captura de texto y tres botones

Figura 4.11. Página web que muestra una ventana de alerta después de haber presionado el primer botón



Eliminación de elementos HTML

Así como se pueden crear nuevos elementos y añadirlos a un documento HTML, también es posible eliminarlos a través de DOM con JavaScript.

Para eliminar un elemento o nodo de un documento HTML, se debe conocer su elemento padre. Para ello se utiliza el método `removeChild()`, el cual se debe ejecutar sobre el elemento padre y recibe como parámetro el elemento hijo que desea eliminarse.

Ejemplo: Eliminando un párrafo de un contenedor

En este ejemplo se realizará un documento HTML para mostrar cómo eliminar un párrafo de un contenedor. El párrafo será eliminado en respuesta al evento `onclick` de un botón, con el cual se disparará la ejecución de una función JavaScript que eliminará el párrafo específico de un contenedor de la página web.

A partir del contenido que se muestra en el listado 4.3, se creará un documento HTML con el nombre *eliminaParrafo.html*; después, se abrirá en un navegador web.

Listado 4.3. Archivo *eliminaParrafo.html*

```
1   <html>
2       <head>
3           <title>Creaci&oacute;n de Elementos HTML</title>
4           <script type="text/javascript">
5               function eliminaParrafo1(){
6                   panel = document.getElementById('panel');
7                   parrafo = document.getElementById('p1');
8                   panel.removeChild(parrafo);
9               }
10          </script>
11      </head>
12      <body>
13          <h2>Eliminaci&oacute;n de Elementos HTML</h2>
14          <p>Para eliminar el p&aacute;rrafo 1
15             da click en el bot&oacute;n:</p>
16          <input type="button" value="Elimina P&aacute;rrafo 1"
17                 onclick="eliminaParrafo1();" />
18          <br /><br />
19          <div id="panel">
20              <p id="p1">Este es el p&aacute;rrafo 1.</p>
21              <p id="p2">Este es el p&aacute;rrafo 2.</p>
22              <p id="p3">Este es el p&aacute;rrafo 3.</p>
23          </div>
24      </body>
25  </html>
```

El listado 4.3 crea un documento HTML con un botón y un `<div>` que contiene tres párrafos. En las líneas 16 y 17 se tiene un botón con el atributo `onclick="eliminaParrafo1();"` indicando que cuando ocurra el evento `onclick` del botón, la función JavaScript `eliminaParrafo1()` será invocada y se encargará de eliminar el párrafo con `id="p1"`. En las líneas 19 a 23 se define un `<div>`, el cual es un contenedor que agrupa los tres párrafos de las líneas 20, 21 y 22, de este contenedor se eliminará el primer párrafo. La función JavaScript `eliminaParrafo1()` que será ejecutada en respuesta al evento `onclick` del botón se encuentra definida de la línea 5 a la 9, en el segmento de código JavaScript entre la etiqueta inicial `<script>` y la etiqueta final `</script>`. La primera línea de la función `eliminaParrafo1()`, en la línea 6 del listado, encuentra al elemento que es el contenedor y almacena una referencia en la variable `panel`; la línea 7 encuentra al elemento que corresponde al primer párrafo y almacena una referencia en la variable `parrafo`; la línea 8 se encarga de eliminar el párrafo del contenedor mediante la ejecución del método `removeChild()`, que

es ejecutado sobre la variable `panel` y toma como parámetro el párrafo que será eliminado.

En este ejemplo, cuando el usuario dé click sobre el botón, se invocará la función JavaScript asociada con el evento `onclick`, la cual eliminará el primer párrafo del contenedor que se encuentra en la página web.

La figura 4.12 muestra la página web *eliminaParrafo.html* de este ejemplo visualizada en un navegador web, donde se observa el botón y los tres párrafos.

La figura 4.13 muestra nuevamente la página web *eliminaParrafo.html* después de que el usuario ha dado click al botón, con lo cual se ha eliminado el primer párrafo del contenedor.

Figura 4.12. Página web *eliminaParrafo.html*

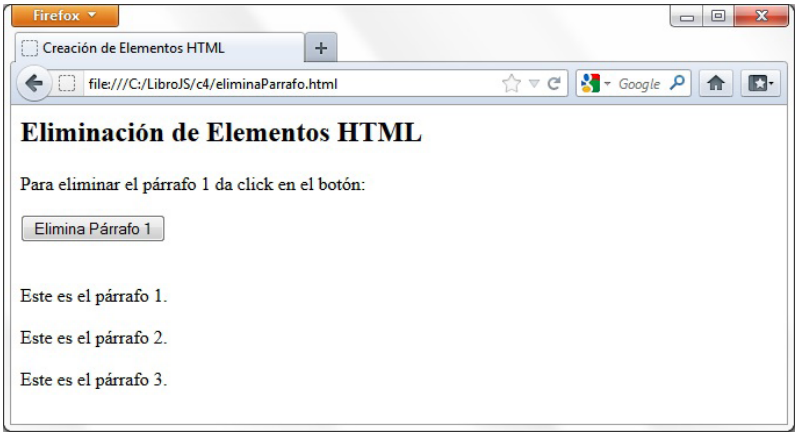
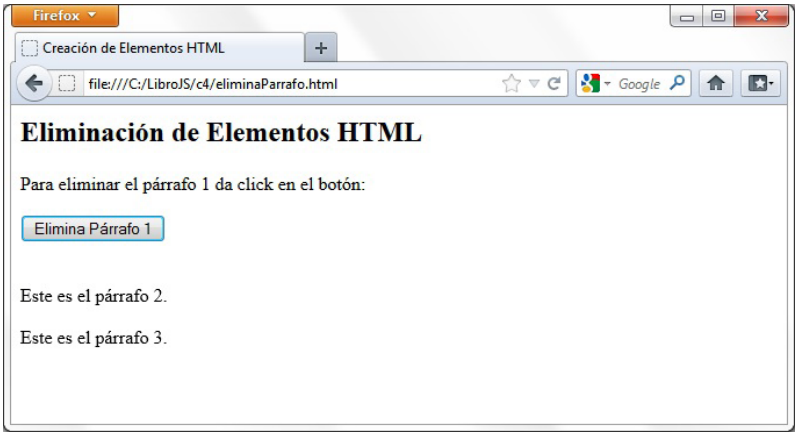


Figura 4.13. Página web después de haber eliminado el primer párrafo



Ejercicio: Eliminando un párrafo específico

Escribe un documento HTML con el nombre *ejercicioEliminaParrafo.html*. El documento debe visualizarse en el navegador web como se muestra en la figura 4.14, en la cual se observa un campo de captura de texto, un botón y tres párrafos. Para este ejercicio puedes tomar como base el ejemplo anterior. En el campo de captura de texto el usuario introducirá el identificador de alguno de los tres párrafos (`p1`, `p2` o `p3`), el cual será utilizado para eliminar el párrafo correspondiente. Por ejemplo, en la figura 4.15 el usuario introdujo `p2` en el campo de captura de texto, por lo que después de dar click en el botón, el segundo párrafo fue eliminado. En la figura 4.16 el usuario introdujo `p1` en el campo de captura de texto, por lo que después de dar click en el botón, el primer párrafo también fue eliminado.

Figura 4.14. Página web *ejercicioEliminaParrafo.html*

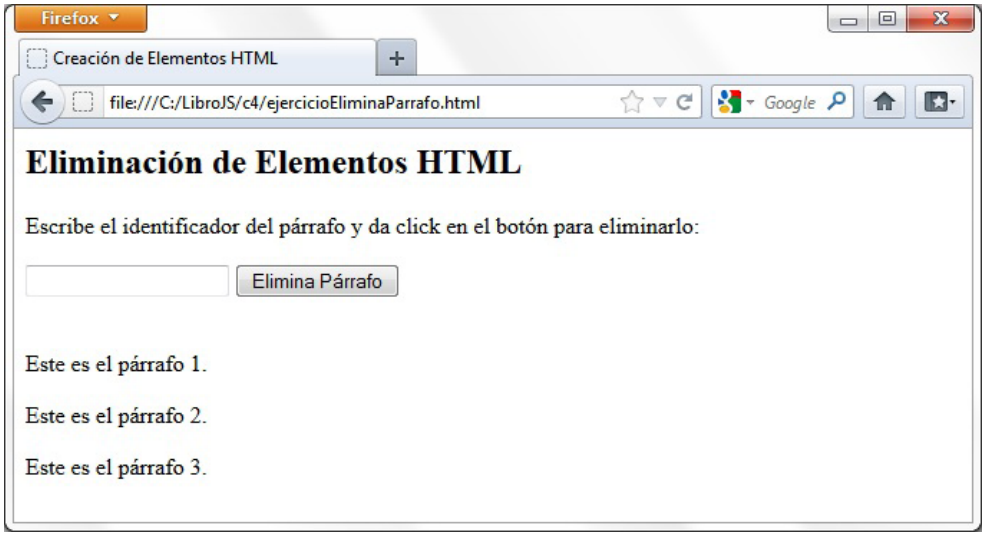


Figura 4.15. Página web después de haber eliminado el segundo párrafo

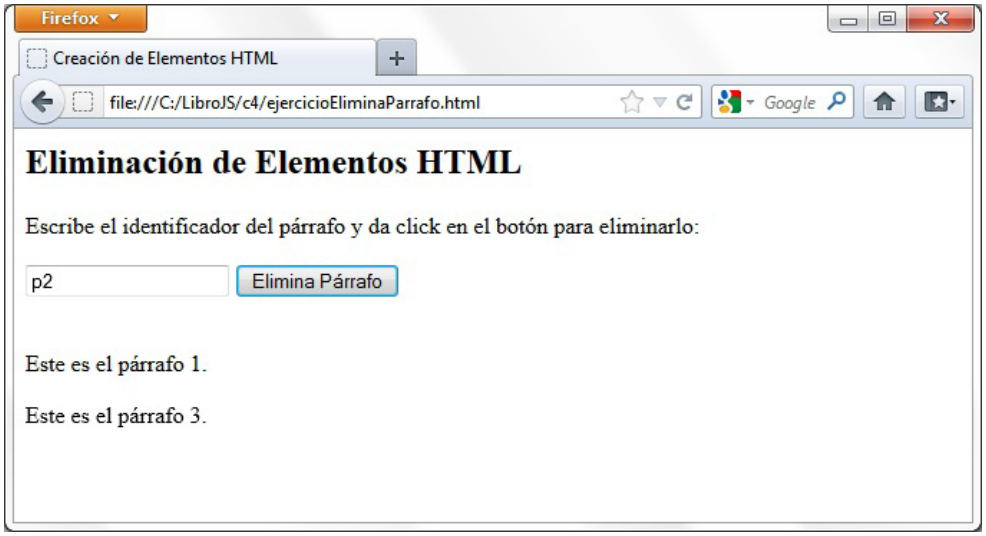


Figura 4.16. Página web después de haber eliminado el primer párrafo

Resumen de métodos

En la tabla 4.1 se presenta un resumen de los métodos revisados en este capítulo.

Tabla 4.1 Métodos del capítulo 4

Método	Descripción
`document.createElement()`	Crea un elemento o nodo. Recibe como parámetro el nombre del elemento
`document.createTextNode()`	Crea un elemento o nodo de texto. Recibe como parámetro una cadena de texto
`setAttribute()`	Crea un atributo. Recibe como parámetros el nombre y el valor del atributo. Se debe ejecutar sobre el elemento donde se creará el atributo
`appendChild()`	Añade un elemento a otro elemento existente. Se ejecuta sobre el elemento existente y recibe como parámetro el elemento a ser añadido
`alert()`	Muestra una ventana emergente. Recibe como parámetro el texto que será mostrado en la ventana

Capítulo 5. Manejo de estilos

En este capítulo se describe cómo cambiar los estilos de los elementos en un documento HTML a través de DOM con JavaScript. Los estilos pueden ser aplicados a cualquiera de los elementos de una página web para modificar su apariencia, tal como el color o el tipo de letra del texto. En este capítulo también se proporcionan varios ejemplos y ejercicios para ilustrar el manejo de estilos de elementos en una página web.

Los objetivos que deben cumplirse al final de este capítulo son:

- Conocer cómo se cambian los estilos de los elementos en una página web.
- Utilizar el atributo *style* de los elementos HTML para cambiar sus atributos.
- Escribir funciones JavaScript para cambiar estilos de elementos en un documento HTML.
- Asociar funciones JavaScript con eventos para cambiar estilos de elementos en un documento HTML.
- Identificar algunos atributos de estilo específicos, como *color, backgroundColor, fontSize, fontFamily, textAlign*, entre otros.

Cambio de estilos a elementos HTML

En un documento HTML es posible cambiar los estilos de sus elementos a través de DOM con JavaScript, lo cual permite de manera dinámica modificar la apariencia de la página web. Los estilos se refieren a todos aquellos atributos o propiedades de las hojas de estilo en cascada (Cascading Style Sheets, CSS, por sus siglas en inglés) que pueden aplicarse a elementos HTML en una página web, como el color de fondo, el color del texto, el tipo de letra del texto, el tamaño de la letra del texto, la alineación del texto, entre otros.

Para cambiar los estilos de un elemento HTML primero se debe acceder al elemento a través del método `getElementById()` del objeto `document` y

después acceder a su atributo `style`, el cual permite tener acceso a todos los atributos de estilos del elemento y, finalmente, acceder al atributo correspondiente y asignarle un valor.

Cambio del color del texto

Uno de los atributos más utilizados es el que controla el color del texto de los elementos HTML que se encuentran en una página web. El atributo `style` de un elemento HTML permite cambiar el atributo `color`, el cual se refiere al color del texto del elemento. Este atributo es aplicable a cualquier elemento HTML que contenga texto, como el cuerpo de una página web `<body>`, los encabezados `<h1>` a `<h6>`, los párrafos `<p>`, las celdas de una tabla `<td>`, entre otros.

Los valores que pueden asignarse a este atributo pueden ser de tres tipos: el nombre del color (`blue`, `red`, `yellow`, `pink`, `orange`, etc.), el valor hexadecimal del color (`#AB567A`, `#56CB4D`, `#B6C788`, etc.) o el valor *red-green-blue* (RGB) del color, como `rgb(200,200,55)`, `rgb(178,67,45)`, `rgb(125,134,90)`, etc.

En la tabla 5.1 se presentan algunos valores comunes que pueden asignarse al atributo `color`. La primera columna muestra el color, la segunda el nombre del color, la tercera el valor hexadecimal y la cuarta el valor RGB. Los valores que pueden asignarse al atributo `color` son los que se encuentran en las columnas segunda, tercera y cuarta.

Tabla 5.1 Nombres de colores, valor hexadecimal y valor RGB

Color	Nombre	Valor Hexadecimal	Valor RGB
amarillo	yellow	#FFFF00	255,255,0
azul	blue	#0000FF	0,0,255
blanco	white	#FFFFFF	255,255,255
café	brown	#A52A2A	165,42,42
gris	gray	#808080	128,128,128
magenta	magenta	#FF00FF	255,0,255
naranja	orange	#FFA500	255,165,0
negro	black	#000000	0,0,0
rojo	red	#FF0000	255,0,0

Color	Nombre	Valor Hexadecimal	Valor RGB
rosa	pink	#FFC0CB	255,192,203
turquesa	cyan	#00FFFF	0,255,255
verde	green	#008000	0,128,0

Ejemplo: Cambiando el color del texto a un párrafo

En este ejemplo se realizará un documento HTML para mostrar cómo cambiar el color del texto de un párrafo. En respuesta al evento `onclick` de un botón, se disparará la ejecución de una función JavaScript que cambiará el color del texto del párrafo.

A partir del contenido que se muestra en el listado 5.1, se creará un documento HTML con el nombre *cambiaColorTextoParrafo.html*; después, se abrirá en un navegador web.

Listado 5.1. Archivo *cambiaColorTextoParrafo.html*

```
<html>
    <head>
        <title>Cambia Estilos de Elementos HTML</title>
        <script type="text/javascript">
            function cambiaTextoAzul(par){
                parrafo = document.getElementById(par);
                parrafo.style.color = "blue";
            }
            function cambiaTextoRojo(par){
                parrafo = document.getElementById(par);
                parrafo.style.color = "red";
            }
        </script>
    </head>
    <body>
        <h2>Cambia Estilos de Elementos HTML</h2>
        <p id="p1">Da click en los botones de abajo para
           cambiar el color del texto de este p&aacute;rrafo.
           El bot&oacute;n con el texto Azul cambiar&aacute;
           el color del texto a azul; y el bot&oacute;n con
           el texto Rojo lo cambiar&aacute; a rojo.</p>
        <br />
        <input type="button" value="Azul"
               onclick="cambiaTextoAzul('p1');" />
        <input type="button" value="Rojo"
               onclick="cambiaTextoRojo('p1');" />
    </body>
</html>
```

El listado 5.1 crea un documento HTML con un párrafo y dos botones. El párrafo se encuentra en las líneas 17 a 21 con su atributo `id="p1"`, el cual será empleado para identificarlo posteriormente. En las líneas 23 y 24 se tiene un botón con el atributo `onclick="cambiaTextoAzul('p1');"` indicando que cuando ocurra el evento `onclick` del botón, la función JavaScript `cambiaTextoAzul()` será invocada, la cual toma el parámetro `p1`, identificador del párrafo al que se le cambiará el color a azul. De manera similar, en las líneas 25 y 26 se tiene un botón con el atributo `onclick="cambiaTextoRojo('p1');"` indicando que cuando ocurra el evento `onclick` del botón, la función JavaScript `cambiaTextoRojo()` será invocada, la cual toma el parámetro `p1`, identificador del párrafo al que se le cambiará el color a rojo. Las funciones JavaScript que serán ejecutadas en respuesta a los eventos `onclick` de los botones están definidas en las líneas 5 a 12, en el segmento de código JavaScript, entre la etiqueta inicial `<script>` y la etiqueta final `</script>`. La función `cambiaTextoAzul()` toma el parámetro `par`, identificador del párrafo al que se le cambiará el color del texto. La primera línea de la función, en la línea 6 del listado, encuentra al elemento párrafo a través de su identificador y almacena una referencia en la variable `parrafo`. La línea 7 se encarga de cambiar el color del texto de dicho párrafo a azul mediante la asignación del valor `"blue"` al atributo `color` del texto del párrafo, el cual se accede a través del atributo `style` del párrafo. La función `cambiaTextoRojo()` toma el parámetro `par`, el cual también representa el identificador del párrafo al que se le cambiará el color del texto. La primera línea de la función, en la línea 10 del listado, encuentra al elemento párrafo a través de su identificador y almacena una referencia en la variable `parrafo`. La línea 11 se encarga de cambiar el color del texto de dicho párrafo a rojo mediante la asignación del valor `"red"` al atributo `color` del texto del párrafo, el cual se accede a través del atributo `style` del párrafo.

En este ejemplo, cada vez que el usuario dé click sobre el botón con el texto *Azul*, se invocará la función JavaScript asociada con el evento `onclick`, la cual cambiará el color del texto del párrafo a azul y, cuando el usuario dé click sobre el botón con el texto *Rojo* se invocará la función JavaScript asociada con el evento `onclick`, la cual cambiará el color del texto del párrafo a rojo.

La figura 5.1 muestra la página web *cambiaColorTextoParrafo.html* de este ejemplo visualizada en un navegador web, donde se observa el párrafo y los dos botones.

La figura 5.2 muestra la página web *cambiaColorTextoParrafo.html* después de que el usuario ha dado click sobre botón con el texto *Azul*, con lo cual se ha cambiado el color del texto del párrafo a azul.

La figura 5.3 muestra la página web *cambiaColorTextoParrafo.html* después de que el usuario ha dado click sobre el botón con el texto *Rojo*, con lo cual se ha cambiado el color del texto del párrafo a rojo.

Figura 5.1. Página web *cambiaColorTextoParrafo.html*

Figura 5.2. Página web después de haber dado click sobre el botón Azul

Figura 5.3. Página web después de haber dado click sobre el botón Rojo

Ejercicio: Cambiando el color de fondo a un párrafo

Escribe un HTML con el nombre *ejercicioCambiaColorFondoParrafo.html*. Este documento debe visualizarse en el navegador web como se muestra en la figura 5.4, en la cual se observa un párrafo y cuatro botones. Para este ejercicio puedes tomar como base el ejemplo anterior. Los primeros dos botones cambian el color del texto del párrafo: el primer botón a azul y el segundo a rojo, como se mostró en el ejemplo anterior. Los otros dos botones se encargan de cambiar el color de fondo del párrafo: el primero a naranja y el segundo a amarillo. La figura 5.5 muestra la misma página web después de que el usuario ha dado click sobre el botón *Azul* y sobre el botón *Fondo Naranja*. La figura 5.6 muestra nuevamente la página web después de que el usuario ha dado click sobre el botón *Rojo* y sobre el botón *Fondo Amarillo*. El atributo `style` que se encarga de cambiar el color de fondo de un elemento HTML es `backgroundColor`.

Figura 5.4. Página web *ejercicioCambiaColorFondoParrafo.html*

Figura 5.5. Página web después de haber dado click sobre el botón Azul y Fondo Naranja

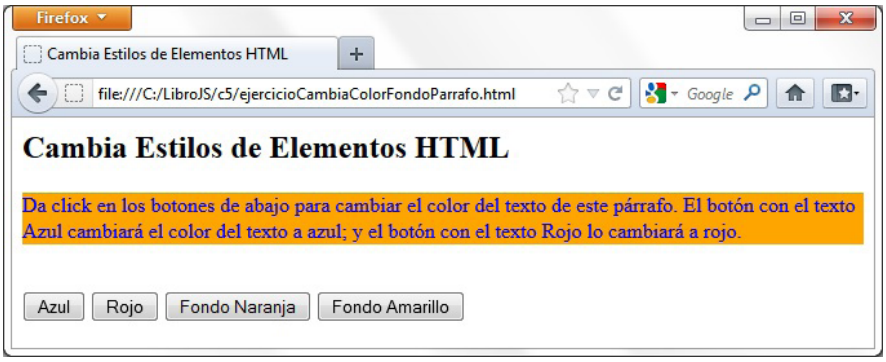
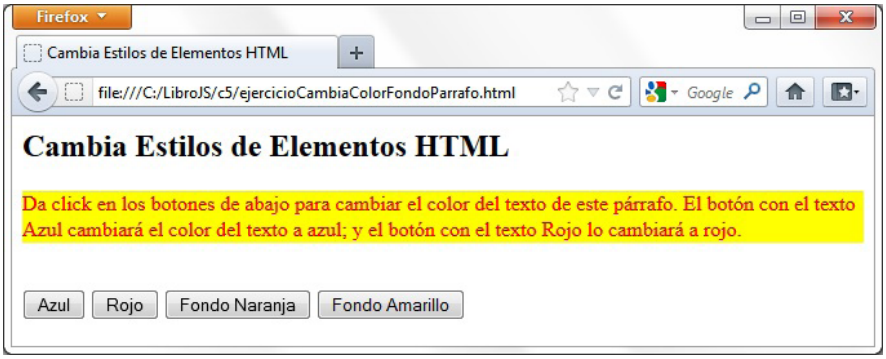


Figura 5.6. Página web después de haber dado click sobre el botón Rojo y Fondo Amarillo



Cambio del tamaño y tipo de letra

Existen atributos que controlan el tamaño y tipo de letra del texto de un elemento HTML dentro de una página web. El atributo `style` de un elemento HTML permite cambiar el atributo `fontSize`, el cual se refiere al tamaño en pixeles de la letra de un elemento, así como el atributo `fontFamily`, el cual hace referencia al tipo de letra del texto de un elemento. Estos atributos son aplicables a cualquier elemento HTML que contenga texto, como el cuerpo de una página web `<body>`, los encabezados `<h1>` a `<h6>`, los párrafos `<p>`, las celdas de una tabla `<td>`, entre otros.

Los valores que pueden asignarse al atributo `fontSize` son números enteros para indicar el tamaño en pixeles, como `10px`, `12px`, `15px`, `20px`, etc., mientras que los valores que pueden asignarse al atributo `fontFamily` son nombres de tipo de letra real o nombres de tipo de letra genérico, como `Arial`, `Times`, `Helvetica`, `Verdana`, `serif`, etc. Los nombres de tipo de letra real deben de existir en el sistema donde se despliegue la página web para que puedan ser visualizados, mientras que los nombres de tipo de letra genérico son una referencia a cualquier tipo de letra instalada en el sistema que sea compatible con el tipo de letra genérico.

En la tabla 5.2 se presentan algunos valores de nombres de tipo de letra real y de nombres de tipo de letra genérico que pueden asignarse al atributo `fontFamily`. La primera columna indica si el tipo de letra es real o genérico y la segunda columna muestra el nombre del tipo de letra. Los valores que pueden asignarse al atributo `fontFamily` son los que se encuentran en la segunda columna.

Tabla 5.2. Tipos de letras y sus nombres

Tipo de letra	Nombre del tipo de letra
Tipo real	Arial
Tipo real	Verdana
Tipo real	Times New Roman
Tipo real	Helvetica
Tipo real	Courier
Tipo real	Georgia
Tipo real	Monaco
Tipo genérico	monospace
Tipo genérico	serif
Tipo genérico	sans-serif
Tipo genérico	cursive
Tipo genérico	fantasy

Ejemplo: Cambiando el tamaño y tipo de letra a un párrafo

En este ejemplo se realizará un documento HTML para mostrar cómo cambiar el tamaño y tipo de letra del texto de un párrafo. En respuesta a eventos `onclick` de varios botones, se disparará la ejecución de diferentes funciones JavaScript que cambiarán el tamaño y tipo de letra del texto del párrafo.

A partir del contenido que se muestra en el listado 5.2, se creará un documento HTML con el nombre *cambiaTipoLetraParrafo.html*; después, se abrirá en un navegador web.

Listado 5.2. Archivo *cambiaTipoLetraParrafo.html*

```
<html>
    <head>
        <title>Cambia Tama&ntilde;o y Tipo de Letra</title>
        <script type="text/javascript">
            function cambia20px(par){
                parrafo = document.getElementById(par);
                parrafo.style.fontSize = "20px";
            }
            function cambia25px(par){
                parrafo = document.getElementById(par);
                parrafo.style.fontSize = "25px";
            }
            function cambiaArial(par){
                parrafo = document.getElementById(par);
                parrafo.style.fontFamily = "Arial";
            }
            function cambiaVerdana(par){
                parrafo = document.getElementById(par);
                parrafo.style.fontFamily = "Verdana";
            }
        </script>
    </head>
    <body>
        <h2>Cambia Tama&ntilde;o y Tipo de Letra</h2>
        <p id="p1">Da click en los botones de abajo para
           cambiar el tama&ntilde;o y tipo de letra
           del texto de este p&aacute;rrafo. Los
           primeros 2 botones cambian el tama&ntilde;o
           del texto, y los últimos 2 botones cambian
           el tipo de letra del texto.</p>
        <br />
        <input type="button" value="20px"
               onclick="cambia20px('p1');" />
        <input type="button" value="25px"
               onclick="cambia25px('p1');" />
        <input type="button" value="Arial"
               onclick="cambiaArial('p1');" />
        <input type="button" value="Verdana"
               onclick="cambiaVerdana('p1');" />
    </body>
</html>
```

El listado 5.2 crea un documento HTML con un párrafo y cuatro botones. El párrafo se encuentra en las líneas 25 a 30, con su atributo `id="p1"`, el cual será empleado para identificarlo posteriormente. En las líneas 32 y 33 se tiene un botón con el atributo `onclick="cambia20px('p1');"` indicando que cuando ocurra el evento `onclick` del botón, la función JavaScript `cambia20px()` será invocada, la cual toma el parámetro `p1`, identificador del párrafo al que se le cambiará el tamaño de letra a 20 pixeles. De manera similar, en las líneas 34 y 35 se tiene un botón con el atributo `onclick="cambia25px('p1');"` indicando que cuando ocurra el evento `onclick` del botón, la función JavaScript `cambia25px()` será invocada, la cual toma el parámetro `p1`, identificador del párrafo al que se le cambiará el tamaño de letra a 25 pixeles. En las líneas 36 y 37 se tiene un botón con el atributo `onclick="cambiaArial('p1');"` indicando que cuando ocurra el evento `onclick` del botón, la función JavaScript `cambiaArial()` será invocada, la cual toma el parámetro `p1`, identificador del párrafo al que se le cambiará el tipo de letra a *Arial*. De manera similar, en las líneas 38 y 39 se tiene un botón con el atributo `onclick="cambiaVerdana('p');"` indicando que cuando ocurra el evento `onclick` del botón, la función JavaScript `cambiaVerdana()` será invocada, la cual toma el parámetro `p1`, identificador del párrafo al que se le cambiará el tipo de letra a *Verdana*. Las funciones JavaScript que serán ejecutadas en respuesta a los eventos `onclick` de los botones están definidas en las líneas 5 a 20, en el segmento de código JavaScript, entre la etiqueta inicial `<script>` y la etiqueta final `</script>`. La función `cambia20px()` toma el parámetro `par`, identificador del párrafo al que se le cambiará el tamaño del texto. La primera línea de la función, en la línea 6 del listado, encuentra al elemento párrafo a través de su identificador y almacena una referencia en la variable `parrafo`. La línea 7 se encarga de cambiar el tamaño de letra del texto de dicho párrafo a 20 pixeles mediante la asignación del valor `"20px"` al atributo `fontSize` del texto del párrafo, el cual se accede a través del atributo `style` del párrafo. La función `cambia25px()` toma el parámetro `par`, identificador del párrafo al que se le cambiará el tamaño de letra del texto. La primera línea de la función, en la línea 10 del listado, encuentra al elemento párrafo a través de su identificador y almacena una referencia en la variable `parrafo`. La línea 11 se encarga de cambiar el tamaño de letra del texto de dicho párrafo a 25 pixeles mediante la asignación del valor `"25px"` al atributo `fontSize` del texto del párrafo, el cual se accede a través del atributo `style` del párrafo. La función `cambiaArial()` toma el parámetro `par`, identificador del párrafo al que se le cambiará el tipo de letra del texto. La primera línea de la

función, en la línea 14 del listado, encuentra al elemento párrafo a través de su identificador y almacena una referencia en la variable `parrafo`. La línea 15 se encarga de cambiar el tipo de letra del texto de dicho párrafo a *Arial* mediante la asignación del valor "`Arial`" al atributo `fontFamily` del texto del párrafo, el cual se accede a través del atributo `style` del párrafo. Finalmente, la función `cambiaVerdana()` toma el parámetro `par`, identificador del párrafo al que se le cambiará el tipo de letra del texto. La primera línea de la función, en la línea 18 del listado, encuentra al elemento párrafo a través de su identificador y almacena una referencia en la variable `parrafo`. La línea 19 se encarga de cambiar el tipo de letra del texto de dicho párrafo a *Verdana* mediante la asignación del valor "`Verdana`" al atributo `fontFamily` del texto del párrafo, el cual se accede a través del atributo `style` del párrafo.

En este ejemplo, cada vez que el usuario dé click sobre el botón con el texto *20px*, se invocará la función JavaScript asociada con el evento `onclick`, la cual cambiará el tamaño de la letra del texto del párrafo a 20 pixeles; cuando el usuario dé click sobre el botón con el texto *25px*, se invocará la función JavaScript asociada con el evento `onclick`, la cual cambiará el tamaño de la letra del texto del párrafo a 25 pixeles; cuando el usuario dé click sobre el botón con el texto *Arial*, se invocará la función JavaScript asociada con el evento `onclick`, la cual cambiará el tipo de letra del texto del párrafo a *Arial*; finalmente, cuando el usuario dé click sobre el botón con el texto *Verdana*, se invocará la función JavaScript asociada con el evento `onclick`, la cual cambiará el tipo de letra del texto del párrafo a *Verdana*.

La figura 5.7 muestra la página web *cambiaTipoLetraParrafo.html* de este ejemplo visualizada en un navegador web, donde se observa el párrafo y los cuatro botones.

La figura 5.8 muestra la página web *cambiaTipoLetraParrafo.html* después de que el usuario ha dado click sobre botón con el texto *20px* y sobre el botón con el texto *Verdana*, con lo cual se ha cambiado el tamaño del texto a 20 pixeles y el tipo de letra del texto a *Verdana*.

La figura 5.9 muestra la página web *cambiaTipoLetraParrafo.html* después de que el usuario ha dado click sobre botón con el texto *25px* y sobre el botón con el texto *Arial*, con lo cual se ha cambiado el tamaño del texto a 25 pixeles y el tipo de letra del texto a *Arial*.

Figura 5.7. Página web *cambiaTipoLetraParrafo.html*

Figura 5.8. Página web después de haber dado click sobre los botones 20px y Verdana

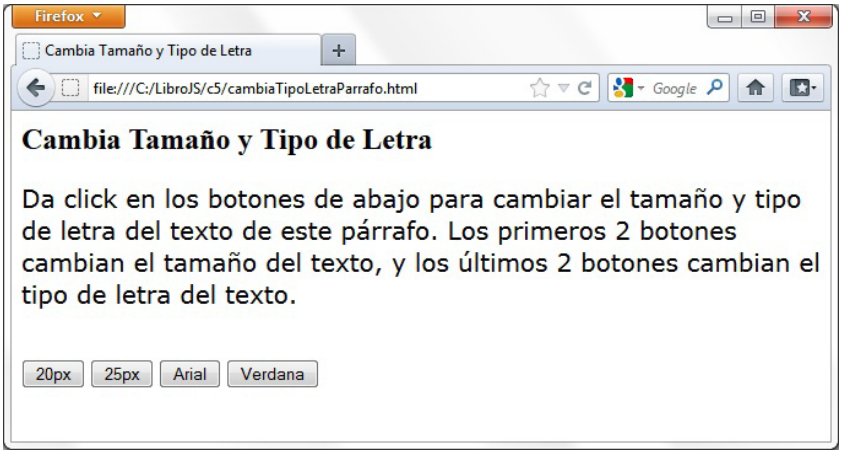
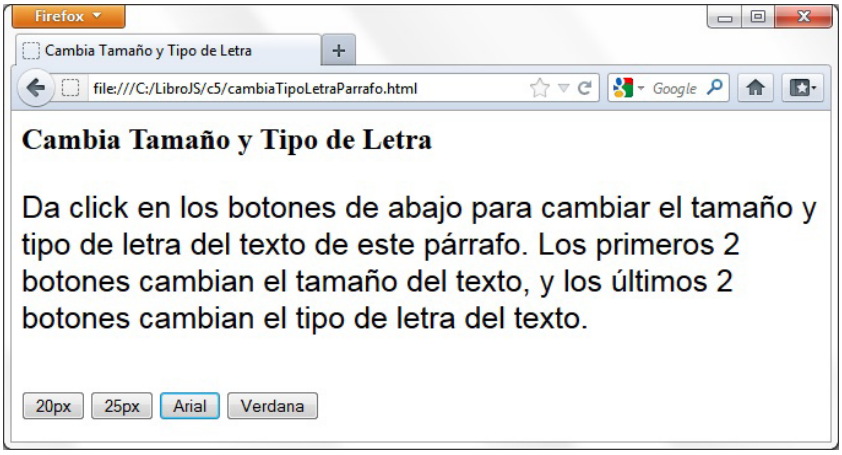


Figura 5.9. Página web después de haber dado click sobre los botones 25px y Arial



Ejercicio: Cambiando la alineación del texto de un párrafo

Escribe un HTML con el nombre *ejercicioCambiaAlineacionParrafo.html*. Este documento debe visualizarse en el navegador web como se muestra en la figura 5.10, en la cual se observa un párrafo y seis botones. Para este ejercicio puedes tomar como base el ejemplo anterior. Los primeros dos botones cambian el tamaño de la letra del párrafo: el primer botón a 20 pixeles y el segundo a 25 pixeles, como se mostró en el ejemplo anterior. Los siguientes dos botones se encargan de cambiar el tipo de letra del párrafo: el primero a *Arial* y el segundo a *Verdana*. Los últimos dos botones cambian la alineación del texto del párrafo: el primero a la derecha y el segundo al centro.

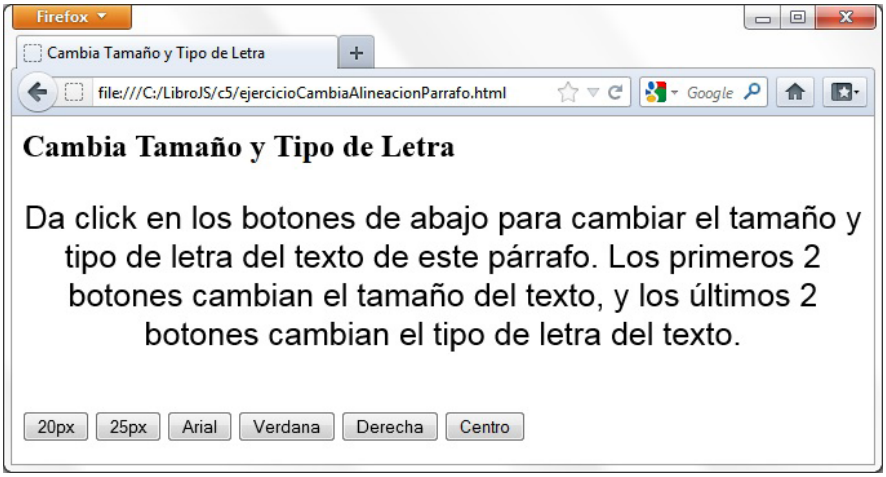
La figura 5.11 muestra la página web *ejercicioCambiaAlineacionParrafo.html* después de que el usuario ha dado click sobre el botón *20px*, sobre el botón *Verdana* y sobre el botón *Derecha*, lo cual modificará el tamaño de la letra del párrafo a 20 pixeles, el tipo de letra del párrafo a *Verdana* y la alineación del texto del párrafo, que será a la derecha.

La figura 5.12 muestra la página web *ejercicioCambiaAlineacionParrafo.html* después de que el usuario ha dado click sobre el botón *25px*, sobre el botón *Arial* y sobre el botón *Centro*, lo cual modificará el tamaño de la letra del párrafo a 25 pixeles, el tipo de letra del párrafo a *Arial* y la alineación del texto del párrafo, que será al centro. El atributo `style` que se encarga de cambiar la alineación del texto de un elemento HTML es `textAlign`.

Figura 5.10. Página web *ejercicioCambiaAlineacionParrafo.html*

Figura 5.11. Página web después de haber dado click sobre los botones 20px, Verdana y Derecha

Figura 5.12. Página web después de haber dado click sobre los botones 25px, Arial y Centro



Resumen de atributos

En la tabla 5.3 se presenta un resumen de los atributos de `style` para cambiar la apariencia de los elementos de una página web revisados en este capítulo. También se incorporan otros atributos importantes que pueden ser utilizados de manera similar a los que se describieron en el capítulo.

Tabla 5.3. Atributos para estilos del capítulo 5

Atributo	Descripción
`backgroundColor`	Color del fondo del texto. Los valores que pueden ser asignados a este atributo son el nombre del color, su valor hexadecimal o su valor RGB. Algunos ejemplos de posibles valores se encuentran en la tabla 5.1
`borderColor`	Color del borde de un elemento. Los valores que pueden ser asignados a este atributo son el nombre del color, su valor hexadecimal o su valor RGB. Algunos ejemplos de posibles valores se encuentran en la Tabla 5.1
`borderStyle`	Estilo del borde de un elemento. Los valores que pueden ser asignados a este atributo son los siguientes: none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset
`borderWidth`	Ancho del borde de un elemento. Los valores que pueden ser asignados a este atributo son valores enteros, que representan el ancho del borde en pixeles
`color`	Color del texto. Los valores que pueden ser asignados a este atributo son el nombre del color, su valor hexadecimal o su valor RGB. Algunos ejemplos de posibles valores se encuentran en la tabla 5.1
`fontFamily`	Tipo de letra del texto. Los valores que pueden ser asignados a este atributo son los tipos de letra soportados por el navegador web. Algunos ejemplos de posibles valores se encuentran en la tabla 5.2
`fontSize`	Tamaño del texto. Los valores que pueden ser asignados a este atributo son valores enteros, que representan el tamaño del texto en pixeles
`fontStyle`	Estilo de la letra del texto. Los valores que pueden ser asignados a este atributo son normal, italic, oblique
`position`	Tipo de método de posicionamiento para un elemento. Los valores que pueden ser asignados a este atributo son static, relative, absolute, fixed
`textAlign`	Alineación del texto. Los valores que pueden ser asignados a este atributo son left, right, center, justify
`visibility`	Indica si un elemento debe estar visible o no
`zIndex`	Indica el orden de superposición de un elemento con respecto a otros elementos en un documento HTML

Bibliografía de consulta y apoyo

Carey, P. *New Perspectives on HTML, XHTML, and Dynamic HTML*, tercera edición, 2005.

Goodman, D. *Dynamic HTML, The Definitive Reference*, segunda edición. O'Reilly, 2002.

González, R., Cordero, J. M. & Valle, J. M. *Diseño de páginas web: iniciación y referencia*. Madrid: Osborne/McGraw Hill, 2001.

King, A. *Optimización de Sitios Web*. Madrid: Anaya Multimedia, 2003.

López, Q. J. *Domine HTML y DHTML*. Ra-Ma, 2003.

Musciano, C. & Kennedy, B. *HTML y XHTML, The Definitive Guide*, quinta edición. O'Reilly, 2002.

Orós, J. C. *Diseño de páginas Web con XHTML, JavaScript y CSS (Navegar en Intenet)*. Ra-Ma, 2006.

Sánchez, G., Santos G. & Molina, M. HTML 4, *Iniciación y Referencia*. Madrid: McGraw Hill, 2001.

Glosario

Atributo: Propiedad o característica que describe a un objeto. Un atributo tiene un nombre y un valor asociado.

Cascading Style Sheets (CSS): Hojas de estilo en cascada, utilizadas para modificar la apariencia de una página web a través de estilos. Los estilos pueden estar contenidos en un archivo separado, en la página web o en el elemento que modifican.

Document Object Model (DOM): Modelo que traduce la estructura de una página web a un árbol de objetos cuando esta es cargada en un navegador web.

Elemento: Nodo del árbol que representa a un documento HTML en DOM, tal como un párrafo, un encabezado, una imagen, una tabla, un contenedor, etc.

Evento: Ocurre en un documento HTML una vez que este está siendo cargado en un navegador web. Cuando un evento ocurre en un documento HTML se puede ejecutar un segmento de código JavaScript como respuesta a tal evento.

Hypertext Markup Language (HTML): Lenguaje de marcado de hipertexto, utilizado para crear páginas web estáticas. HTML es un lenguaje basado en etiquetas que define la estructura y contenido de una página web.

JavaScript: Lenguaje script basado en objetos, diseñado específicamente para hacer que las páginas web sean dinámicas e interactivas. JavaScript es un lenguaje para hacer programación web dinámica del lado del cliente.

Método: Comportamiento o acción que un objeto puede ejecutar. Un método tiene un nombre y un conjunto de parámetros.

Objeto: Un objeto en JavaScript es un elemento que proviene de un documento HTML a partir de que el documento se ha cargado en un navegador web. Un objeto contiene atributos y métodos.

Acerca del autor

Carlos Roberto Jaimez González es profesor investigador del Departamento de Tecnologías de la Información de la Universidad Autónoma Metropolitana Unidad Cuajimalpa. Es doctor en Ciencias de la Computación por la Universidad de Essex, Reino Unido, maestro en Tecnologías de Comercio Electrónico por la misma universidad y licenciado en Computación por la Universidad Autónoma Metropolitana Unidad Iztapalapa.

En el campo docente ha impartido cursos a nivel licenciatura y posgrado en la Universidad de Essex y en la Universidad Autónoma Metropolitana, tales como Programación de Web Estático, Programación de Web Dinámico, Programación Orientada a Objetos, Programación de Aplicaciones Web, Tecnologías Web, Integración de Sistemas, Bases de Datos, Programación de Aplicaciones para Comercio Electrónico, Análisis y Diseño de Sistemas, entre otras. La Universidad Autónoma Metropolitana le ha otorgado en dos ocasiones el Premio a la Docencia (2011 y 2014) en reconocimiento a su labor docente destacada y en particular por la producción de materiales didácticos de calidad.

En el área de investigación, es autor de publicaciones en revistas especializadas y congresos nacionales e internacionales sobre temas en tecnologías y sistemas para apoyar la educación, interoperabilidad en sistemas distribuidos, objetos distribuidos y servicios web, aplicaciones web y móviles para comercio electrónico, así como sistemas multiagente.

En la industria ha participado en diversos proyectos en organizaciones nacionales e internacionales, desempeñándose como desarrollador de sistemas, administrador de bases de datos, líder de proyectos, consultor e instructor certificado en tecnologías de la información.

Su sitio web puede ser consultado en: http://ccd.cua.uam.mx/~cjaimez/

Programación de Web Dinámico se terminó de imprimir en julio de 2015 de forma digital en los talleres de Imprenta 1200+ Andorra 29. Colonia Del Carmen Zacahuitzco, México D.F. Tel. (52)55218493.

El tiraje consta de 100 ejemplares de 17x24 cm, 88 páginas cada uno, a cuatro tintas, encuadernación pegado cubierta flexible. En su composición se utilizó la familia Avenir. Se empleó papel reciclado de 90g para páginas interiores.