Materia: Tecnologías para la Web.
Tema: JavaScript: Modelo de objetos de documento (DOM).

1. Introducción

- El Modelo de objetos de documento nos brinda acceso a todos los elementos en una página Web. Mediante el uso de JavaScript es posible crear, modificar y eliminar en forma dinámica los elementos en la página.

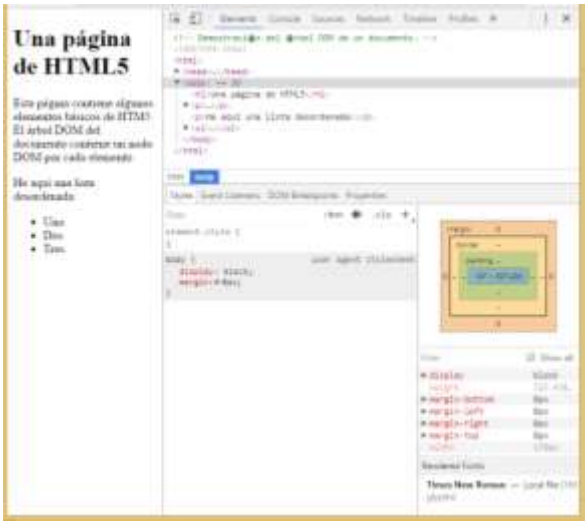
2. Modelado de un documento: nodos y árboles DOM

- El método `getElementById` devuelve objetos conocidos como nodos DOM. Cada elemento en una página de HTML5 se modela en el navegador Web mediante un nodo DOM.
- Todos los nodos en un documento componen el árbol DOM de la página Web, el cual describe las relaciones entre los elementos.
- Los nodos se relacionan entre sí por medio de relaciones hijo-padre. Se dice que un elemento de HTML5 dentro de otro elemento es su hijo: el elemento contenedor se conoce como el padre. Un nodo puede tener varios hijos peto sólo un padre. Los nodos con d mismo nodo padre se conocen como hermanos.
- El nodo del documento en un árbol DOM se conoce como nodo raíz, ya que no tiene padre.

Navegador	Comando para mostrar las herramientas de desarrollador
Chrome	*Windows/Linux: Control + Mayús + i* *Mac OS X: Comando + Opción + i*
Firefox	*Windows/Linux: Control + Mayús + i* *Mac OS X: Comando + Mayús + i*
Internet Explorer	*F12*
Opera	*Windows/Linux: Control + Mayús + i* *Mac OS X: Comando + Opción + i*
Safari	*Windows/Linux: Control + Mayús + i* *Mac OS X: Comando + Opción + i*

EJEMPLO 1.
Archivo `arboldom.html`:

```
<!-- Demostración del árbol DOM de un documento. -->
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>Demostraci&oacute;n de un &aacute;rbol DOM</title>
	</head>
	<body>
		<h1>Una p&aacute;gina de HTML5</h1>
		<p>Esta p&aacute;gina contiene algunos elementos b&aacute;sicos de HTM5. El
&aacute;rbol DOM
			del documento contiene un nodo DOM por cada elemento</p>
		<p>He aqu&iacute; una lista desordenada:</p>
		<ul>
			<li>Uno</li>
			<li>Dos</li>
			<li>Tres</li>
		</ul>
	</body>
</html>
```



3. Recorrido y modificación de un árbol de DOM

- Los métodos `setAttribute` y `getAttribute` de los elementos del DOM nos permiten modificar el valor de un atributo y obtener el valor del atributo de un elemento, respectivamente.
- El método `createElement` del objeto `document` crea un nuevo nodo DOM y recibe el nombre de la etiqueta como argumento. Cabe mencionar que, aunque `createElement` crea un nuevo elemento, no inserta el elemento en la página.
- El método `createTextNode` de `document` crea un nodo DOM que puede contener sólo texto. Dado un argumento de cadena, `createTextNode` inserta la cadena en el nodo de texto.
- El método `appendChild` se llama en un nodo padre para insertar un nodo hijo (se pasa como argumento) después de cualquier hijo existente.
- La propiedad `nodoPadre` de cualquier nodo DOM contiene el padre del nodo.
- El método `insertBefore` se llama en un padre, recibe un nuevo hijo y un hijo existente como argumentos.
- El nuevo hijo se inserta como hijo del padre justo antes del hijo existente.
- El método `replaceChild` se llama en un nodo padre, recibe un nuevo hijo y un hijo existente como argumentos. El método inserta el nuevo hijo en su lista de hijos en lugar del hijo existente.
- El método `removeChild` se llama en un nodo padre y recibe como argumento un hijo que se desea eliminar.

EJEMPLO 2.
Archivo dom.html:

```
<!-- Funcionalidad básica del DOM. -->
<!DOCTYPE html>
<html>
      <head>
            <meta charset = "utf-8">
            <title>Funcionalidad b&aacute;sica del DOM</title>
            <link rel = "stylesheet" type = "text/css" href = "estilo.css">
            <script src = "dom.js" ></script>
      </head>
      <body>
            <h1 id = "encabezadogrande" class = "highlighted">
                  [encabezadogrande] Modelo de objetos DHTML</h1>
            <h3 id = "encabezadochico">[encabezadochico] Funcionalidad de elementos</h3>
            <p id = "para1">[para1] El modelo de objetos de documento (DOM) permite
            el acceso r&aacute;pido y din&aacute;mico a todos los elementos en un documento
de HTML5
                  para manipularlos con JavaScript.</p>
```

```
            <p id = "para2">[para2] Para mayor informaci&oacute;n, revise la
                  secci&oacute;n "JavaScript y el DOM" del
                  <a id = "link" href = "http://www.deitel.com/javascript">
                        [link] Centro de recursos de JavaScript</a> de Deitel.</p>
            <p id = "para3">[para3] Los siguientes botones demuestran:(lista)</p>
            <ul id = "lista">
                  <li id = "elemento1">[elemento1] getElementById y parentNode</li>
                  <li id = "elemento2">[elemento2] insertBefore y appendChild</li>
                  <li id = "elemento3">[elemento3] replaceChild y removeChild</li>
            </ul>
            <div id = "nav" class = "nav">
                  <form onsubmit = "regresa falso" action = "#">
                        <p><input type = "text" id = "gbi" value = "encabezadogrande">
                              <input type = "button" value = "Obtener por id"
                                    id = "botonPorId"></p>
                        <p><input type = "text" id = "ins">
                              <input type = "button" value = "Insertar antes"
                                    id = "botonInsertar" ></p>
                        <p><input type = "text" id = "adjuntar">
                              <input type = "button" value = "Adjuntar hijo"
                                    id = "botonAdjuntar"></p>
                        <p><input type = "text" id = "reemplazar">
                              <input type = "button" value = "Reemplazar actual"
                                    id = "botonReemplazar"></p>
                        <p><input type = "button" value = "Eliminar actual"
                              id = "botonEliminar"></p>
                        <p><input type = "button" value = "Obtener padre"
                              id = "botonPadre"></p>
                  </form>
            </div>
      </body>
</html>
```

Archivo dom.js:

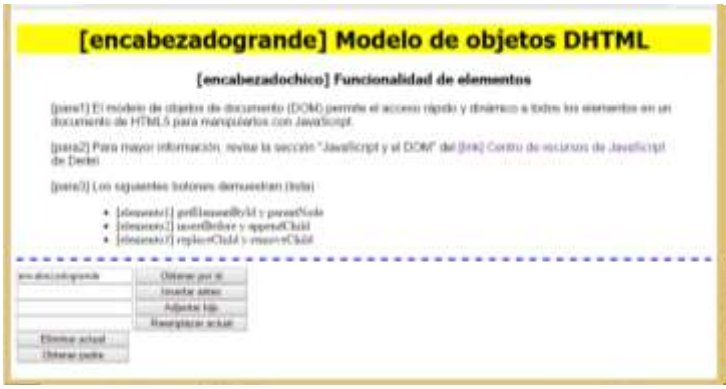
```
// Secuencia de comandos para demostrar la funcionalidad básica del DOM.
var nodoActual; // almacena el nodo actual resaltado
var cuentaid = 0; // se usa para asignar un id único a los nuevos elementos
// registrar manejadores de eventos e inicializar nodoActual
function iniciar(){
      document.getElementById( "botonPorId" ).addEventListener(
            "click", porId, false ) ;
      document.getElementById( "botonInsertar" ).addEventListener(
            "click", insertar, false);
      document.getElementById( "botonAdjuntar" ).addEventListener(
            "click", adjuntarNodo, false);
      document.getElementById( "botonReemplazar" ).addEventListener(
            "click", reemplazarActual, false );
      document.getElementById( "botonEliminar" ).addEventListener(
            "click", eliminar, false );
      document.getElementById( "botonPadre" ).addEventListener(
            "click", padre, false );
      // inicializar nodoActual
      nodoActual = document.getElementById( "encabezadogrande" );
} // fin de la función iniciar
// llamar a iniciar después de que se carga la ventana
window.addEventListener( "load", iniciar, false );
// obtener y resaltar un elemento en base a su atributo id
```

```
function porId(){
      var id = document.getElementById( "gbi" ).value;
      var destino = document.getElementById( id );
      if ( destino )
            cambiarA( destino );
} // fin de la función porId
// insertar un elemento párrafo antes del elemento actual
// usado el método insertBefore
function insertar(){
      var nuevoNodo = crearNuevoNodo(
            document.getElementById( "ins" ).value );
      nodoActual.parentNode.insertBefore( nuevoNodo, nodoActual );
      cambiarA( nuevoNodo );
} // fin de la función insertar
// adjuntar un nodo párrafo como hijo del nodo actual
function adjuntarNodo(){
      var nuevoNodo = crearNuevoNodo(
            document.getElementById( "adjuntar" ).value );
      nodoActual.appendChild( nuevoNodo );
      cambiarA( nuevoNodo );
} // fin de la función adjuntarNodo
// reemplaza el nodo actual seleccionado con un nodo párrafo
function reemplazarActual(){
      var nuevoNodo = crearNuevoNodo(
            document.getElementById( "reemplazar" ).value );
      nodoActual.parentNode.replaceChild( nuevoNodo, nodoActual );
      cambiarA( nuevoNodo );
} // fin de la función reemplazarActual
// eliminar el nodo actual
function eliminar(){
      if ( nodoActual.parentNode == document.body )
            alert( "No se puede eliminar un elemento de nivel superior." );
      else{
            var nodoAnterior = nodoActual;
            cambiarA( nodoAnterior.parentNode );
            nodoActual.removeChild( nodoAnterior );
      }
} // fin de la función eliminar
// obtener y resaltar el padre del nodo actual
function padre(){
      var destino = nodoActual.parentNode;
      if ( destino != document.body )
            cambiarA( destino );
      else
            alert( "No hay padre." );
} // fin de la función padre
// función ayudante que devuelve un nuevo nodo párrafo que contiene
// un id único y el texto dado
function crearNuevoNodo( texto ){
      var nuevoNodo = document.createElement( "p" );
      idNodo = "nuevo" + cuentaid;
      ++cuentaid;
      nuevoNodo.setAttribute( "id", idNodo ); // establecer id de nuevoNodo
      texto = "[" + idNodo + "] " + texto;
      nuevoNodo.appendChild( document.createTextNode( texto ) );
      return nuevoNodo;
} // fin de la función crearNuevoNodo
```

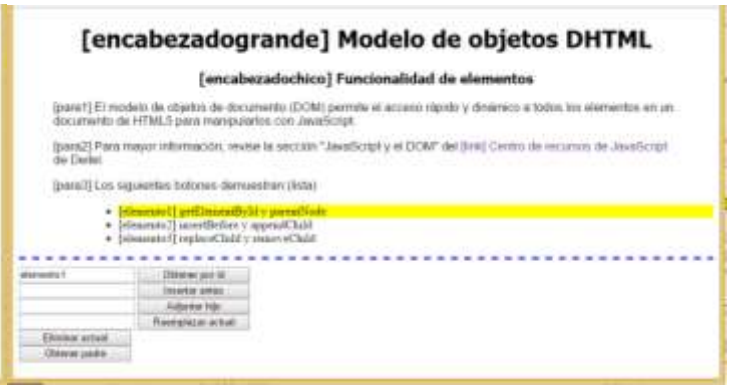
```
// función ayudante que cambia a un nuevo nodoActual
function cambiarA( nuevoNodo ){
      nodoActual.setAttribute( "class", "" ); // elimina el resaltado anterior
      nodoActual = nuevoNodo;
      nodoActual.setAttribute( "class", "highlighted" ); // resaltar
      document.getElementById( "gbi" ).value =
            nodoActual.getAttribute( "id" );
} // fin de la función cambiarA
```

Archivo estilo.css:

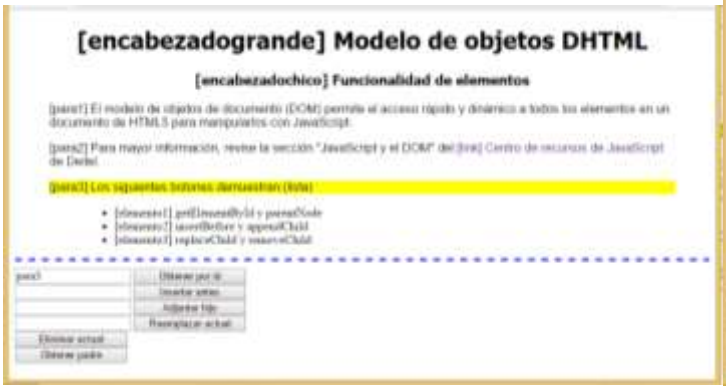
```
/* CSS para dom.html. */
h1, h3      { text-align: center;
              font-family: tahoma, geneva, sans-serif; }
p           { margin-left: 5%;
              margin-right: 5%;
              font-family: arial, helvetica, sans-serif; }
ul          { margin-left: 10%; }
a           { text-decoration: none; }
a:hover     { text-decoration: underline; }
.nav        { width: 100%;
              border-top: 3px dashed blue;
              padding-top: 10px; }
.highlighted { background-color: yellow; }
input       { width: 150px; }
form > p    { margin: 0px; }
```



Documento inicial.

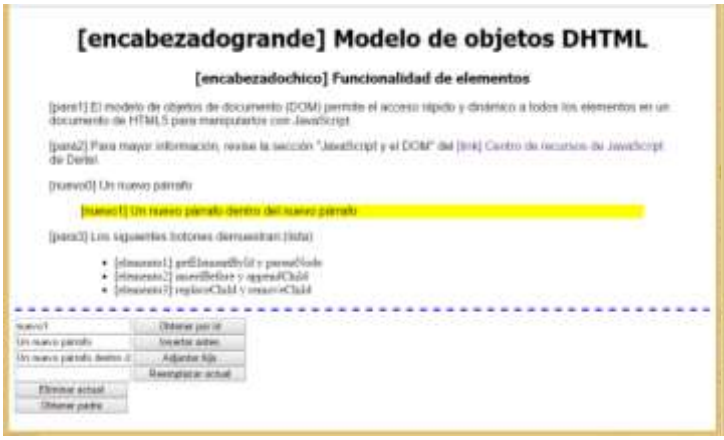


Ingresar `elemento1` y digitar botón Obtener por id.

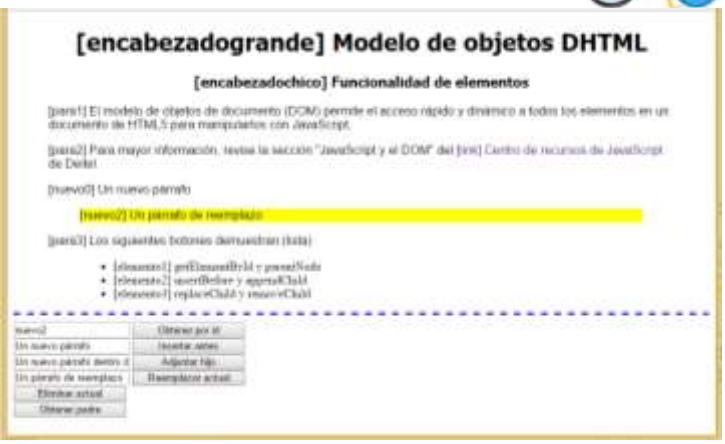


Ingresar `para3` y digitar botón Obtener por id.

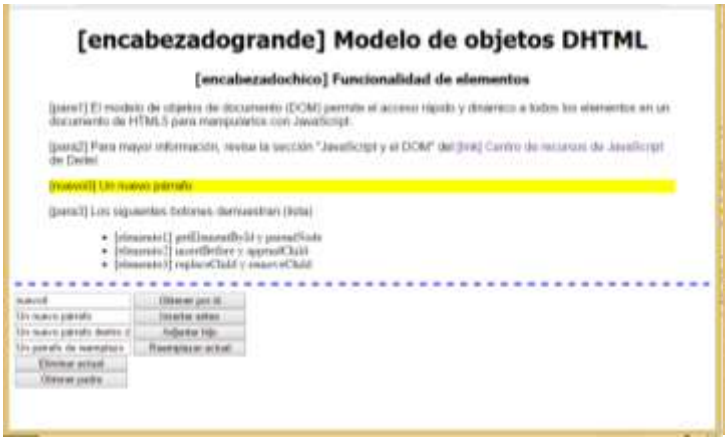
Digitar botón Insertar antes.



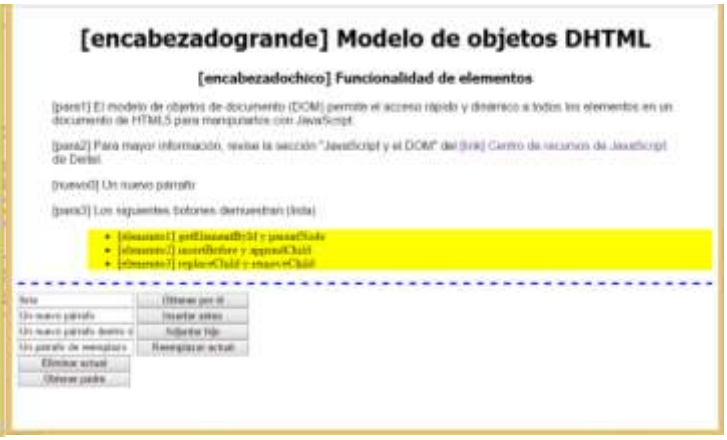
Digitar botón Adjuntar hijo.



Digitar botón Reemplazar actual.



Digitar botón Eliminar actual.



Ingresar `elemento2` y digitar botón Obtener por id, luego digitar botón Obtener padre.

4. Colecciones del DOM

- El DOM contiene varias colecciones que son grupos de objetos relacionados en una página. Las colecciones del DOM se utilizan como propiedades de objetos DOM, como el objeto `document` o un nodo DOM.
- El objeto `docunent` tiene propiedades que contienen la colección `images`, la colección links, la colección `forms` y la colección `anchors`. Estas colecciones contienen todos los elementos del tipo correspondiente en la página.
- Para buscar el número de elementos en la colección, use la propiedad `length` de ésta.
- Para acceder a los elementos en una colección, use el método `item`, o use corchetes igual que como con un arreglo. El método `item` de una colección DOM se utiliza para acceder a los elementos específicos en una colección y recibe un índice como argumento. El método `namedItem` recibe un nombre como parámetro y busca el elemento en la colección, si lo hay, cuyo atributo `id` o `name` coincida con ese nombre.
- La propiedad `href` de un nodo de vínculo de DOM se refiere al atributo `href` del vínculo.

EJEMPLO 3.
Archivo colecciones.html:

```
<!-- Uso de la colección links. -->
<!DOCTYPE html>
<html>
      <head>
            <meta charset = "utf-8">
            <title>Uso de la colecci&oacute;n links</title>
```

```
            <link rel = "stylesheet" type = "text/css" href = "estilo.css">
            <script src = "colecciones.js" ></script>
      </head>
      <body>
            <h1>Centros de recursos de Deitel</h1>
            <p><a href = "http://www.deitel.com/"> El sitio Web de Deitel</a>
                  contiene una creciente
                  <a href = "http://www.deitel.com/ResourceCenters.html">lista
                        de Centros de recursos</a> sobre un amplio rango de temas. Muchos
                        centros de recursos se relacionan con los temas cubiertos en este
libro,
                  <a href = "http://www.deitel.com/books/iw3htp5">C&oacute;mo programar
                        en Internet y World Wide Web, 5ta edici&oacute;n</a>. Tenemos
                        centros de recursos sobre
                  <a href = "http://www.deitel.com/Web 2.0">Web 2.0</a>,
                  <a href = "http://www.deitel.com/Firefox">Firefox</a> e
                  <a href = "http://www.deitel.com/IE9">Internet Explorer 9</a>,
                  <a href = "http://www.deitel.con/HTML5">HTML5</a>, y
                  <a href = "http://www.deitel.com/JavaScript">JavaScript</a>.
                  Est&eacute; atento a los nuevos centros de recursos relacionados.</p>
            <p>V&iacute;nculos en esta p&aacute;gina:</p>
            <div id = "vinculos"></div>
      </body>
</html>
```

Archivo colecciones.js:

```
// Secuencia de comandos para demostrar el uso de la colección links.
function procesarVinculos(){
      var listaVinculos = document.links; // obtiene los vínculos del documento
      var contenido = "<ul>";
      // concatenar cada vinculo a contenido
      for ( var i = 0; i < listaVinculos.length; ++i ){
            var vinculoActual = listaVinculos[ i ];
            contenido += "<li><a href='" + vinculoActual.href + "'>" +
                  vinculoActual.innerHTML + "</li>";
      } // fin de for
      contenido += "</ul>";
      document.getElementById( "vinculos" ).innerHTML = contenido;
} // fin de la función procesarVinculos
window.addEventListener( "load", procesarVinculos, false );
```

Archivo estilo.css:

```
/* CSS para colecciones.html. */
body        { font-family: arial, helvetica, sans-serif; }
h1          { font-fam ily: tahoma, geneva, sans-serif;
                  text-align: center; }
p a         { color: DarkRed; }
ul          { font-size: .9em; }
li          { display: inline;
                  list-style-type: none;
                  border-right: 1px solid gray;
                  padding-left: 5px; padding-right: 5px; }
li:first-child    { padding-left: 0px; }
li:last-child     { border-right: none; }
a           { text-decoration: none; }
a:hover     { text-decoration: underline; }
```

5. Estilos dinámicos

- El estilo de un elemento puede cambiarse en forma dinámica. A menudo dicho cambio se hace en respuesta a los eventos del usuario. Tales cambios de estilo pueden crear muchos efectos, incluyendo efectos de pasar el ratón sobre un elemento en la pantalla, menús interactivos y animaciones.
- La propiedad `body` del objeto `document` se refiere al elemento `body` en la página de HTML5.
- El método `setInterval` del objeto `window` ejecuta en forma repetida una instrucción en cierto intervalo. Recibe dos parámetros: una instrucción para ejecutarse repetidas veces y un entero que especifica con qué frecuencia ejecutarla, en milisegundos. El método `setInterval` devuelve un identificador único para llevar el registro de ese intervalo específico.
- El método `clearInterval` del objeto `window` detiene las llamadas repetitivas del método `setInterval` del objeto. A `clearInterval` le pasamos el identificador de intervalo que devuelve `setInterval`.

EJEMPLO 4.

Archivo estilodinamico.html:

```
<!-- Estilos din&aacute;micos. -->
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>Estilos din&aacute;micos</title>
		<script src = "estilodinamico.js" ></script>
	</head>
	<body>
		<p>Bienvenido a nuestro sitio Web.</p>
	</body>
</html>
```

Archivo estilodinamico.js:

```
// Secuencia de comandos para demostrar los estilos dinámicos.
function iniciar(){
	var colorEntrada = prompt( "Escriba el nombre de un color para el " +
		"fondo de esta página", "" ) ;
	document.body.setAttribute( "style",
		"background-color: " + colorEntrada );
} // fin de la función iniciar
window.addEventListener( "load", iniciar, false );
```

EJEMPLO 5.

Archivo visorportadas.html:

```
<!-- Estilos dinámicos utilizados para animación. -->
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>Visor de portadas de libros de Deitel</title>
		<link rel = "stylesheet" type = "text/css" href = "estilo.css">
		<script src = "visorportadas.js" ></script>
	</head>
	<body>
		<div id = "imgprincip">
			<img id = = "imgPortada" src = "normal/jhtp.jpg"
				alt = "Imagen de portada normal">
		</div>
		<div id = "miniaturas">
			<img src = "miniaturas/jhtp.jpg" id = "jhtp"
				alt = "Portada de C&oacute;no programar en Java">
			<img src = "miniaturas/iw3htp.jpg" id = "iw3htp"
				alt = "Portada de C&oacute;no programar en Internet y World Wide
Web">
			<img src = "miniaturas/cpphtp.jpg" id = "cpphtp"
				alt = "Portada de C&oacute;mo programar en C++">
			<img src = "miniaturas/jhtplov.jpg" id = "jhtplov"
				alt = "Portada de C&oacute;no programar en Java LOV">
			<img src = "miniaturas/cpphtplov.jpg" id = "cpphtplov"
				alt = "Portada de C&oacute;mo programar en C++ LOV">
			<img src = "miniaturas/vcsharphtp.jpg" id = "vcsharphtp"
				alt = "Portada de C&oacute;mo programar en Visual C#">
		</div>
	</body>
</html>
```

Archivo visorportadas.js:

```
// Secuencia de comandos para demostrar los estilos dinámicos que se utilizan en animaciones.
var intervalo = null; // lleva el registro del intervalo
var velocidad = 6; // determina la velocidad de la animación
var cuenta = 0; // tamaño de la imagen durante la animación
// se llama repetidas veces para animar la portada del libro
function ejecutar(){
	cuenta += velocidad;
	// detener la animación cuando la imagen sea lo bastante grande
	if ( cuenta >= 375 ){
		window.clearInterval( intervalo );
		intervalo = null;
	} // fin de if
	var imagenGrande = document.getElementById( "imgPortada" );
	imagenGrande.setAttribute( "style", "width: " + (0.7656 * cuenta + "px;") +
```

```
            "height: " + (cuenta + "px;") );
} // fin de la función ejecutar
// inserta la imagen apropiada en el área de la imagen principal y comienza la animación
function mostrar( archimg ){
      if ( intervalo )
            return;
      var imagenGrande = document.getElementById( "imgPortada" );
      imagenGrande.setAttribute( "style", "width: 0px; height: 0px;" );
      imagenGrande.setAttribute( "src", "normal/" + archimg );
      imagenGrande.setAttributeC "alt", "Versi&oacute;n grande de " + archimg );
      cuenta = 0; // iniciar la imagen en tamaño 0
      intervalo = window.setInterval( "ejecutar()", 10 ); // animar
} // fin de la función mostrar
// registrar manejadores de eventos
function iniciar(){
      document.getElementById( "jhtp" ).addEventListener(
            "click", function() { mostrar( "jhtp.jpg" ); }, false );
      document.getElementById( "iw3htp" ).addEventListener(
            "click", function() { mostrar( "iw3htp.jpg" ); }, false );
      document.getElementById( "cpphtp" ).addEventListener(
            "click", function() { mostrar( "cpphtp.jpg" ); }, false);
      document.getElementById( "jhtplov" ).addEventListener(
            "click", function() { mostrar( "jhtplov.jpg" ); }, false );
      document.getElementById( "cpphtplov" ).addEventListener(
            "click", function() { mostrar( "cpphtplov.jpg" ); }, false );
      document.getElementById( "vcsharphtp" ).addEventListener(
            "click", function() { mostrar( "vcsharphtp.jpg" ); }, false );
} // fin de la función iniciar
window.addEventListener( "load", iniciar, false );
```

Archivo estilo.css:

```
/* CSS para visor portadas.html. */
#miniaturas { width: 192px;
                  height: 370px;
                  padding: 5px;
                  float: left; }
#imgprincip { width: 389px;
                  padding: 5px;
                  float: left; }
#imgPortada { height: 373px; }
img         { border: 1px solid black; }
```


Java
HOW TO PROGRAM
PAUL DEITEL
HARVEY DEITEL


Java
HOW TO PROGRAM
PAUL DEITEL
HARVEY DEITEL