Internet & World Wide Web: How to Program by Deitel and Deitel

Document Object Model (DOM): Objects and Collections Chapter 12

1

OBJECTIVES

- In this chapter you will learn:
  - How to use JavaScript and the W3C Document Object Model to create dynamic web pages.
  - The concept of DOM nodes and DOM trees.
  - How to traverse, edit and modify elements in an XHTML document.
  - How to change CSS styles dynamically.
  - To create JavaScript animations.

2

Chapter 12 Sections

- 12.1 Introduction
- 12.2 Modeling a Document: DOM Nodes and Trees
- 12.3 Traversing and Modifying a DOM Tree
- 12.4 DOM Collections
- 12.5 Dynamic Styles
- 12.6 Summary of the DOM Objects and Collections
- 12.7 Wrap-Up
- 12.8 Web Resources

3

What is DOM?

- The World Wide Web Consortium (W3C) defines the Document Object Model (DOM)
- Platform- and language-neutral
- Permits script to access and update the content, structure, and style of a document

How Do Web Authors Use the DOM?

- To access everything in the web page document
  - Make numerous content updates
  - Work with content in separate document fragments
- Working together with the Dynamic HTML (DHTML) Object Model

DOM Advantages

- Enhances a Web author's ability to build and manage complex documents and data
- Moving an object from one part of the document to another is easy

DOM and Dynamic Effects

- XHTML elements can be treated as objects with the DOM
- Attributes of XHTML elements can be treated as properties of those objects
- Thus, scripting can be used to address objects through their id attribute, and property values (attributes) can be changed at runtime with JavaScript
- Result = “dynamic” effects

7

DHTML and DOM

- Microsoft’s DHTML Object Model provides access to almost all document elements and to all attributes of an element
- W3C DOM is consistent with the DHTML Object Model in that every element and every attribute is accessible in script
- W3C DOM is ant evolution from the DHTML Object Model

Fig. 12.1 | Demonstration of a document’s DOM tree (Part 1 of 4).

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 12.1: domtree.html -->
<!-- Demonstration of a document's DOM tree. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>DOM Tree Demonstration</title>
   </head>
   <body>
      <h1>An XHTML Page</h1>
      <p>This page contains some basic XHTML elements. We use the Firefox
         DOM Inspector and the IE Developer Toolbar to view the DOM tree
         of the document, which contains a DOM node for every element in
         the document.</p>
      <p>Here's a list:</p>
      <ul>
         <li>One</li>
         <li>Two</li>
         <li>Three</li>
      </ul>
   </body>
</html>
```

9

Fig. 12.1 | Demonstration of a document's DOM tree (Part 2 of 4).

10

Fig. 12.1 | Demonstration of a document's DOM tree (Part 3 of 4).

11

Fig. 12.1 | Demonstration of a document's DOM tree (Part 4 of 4).

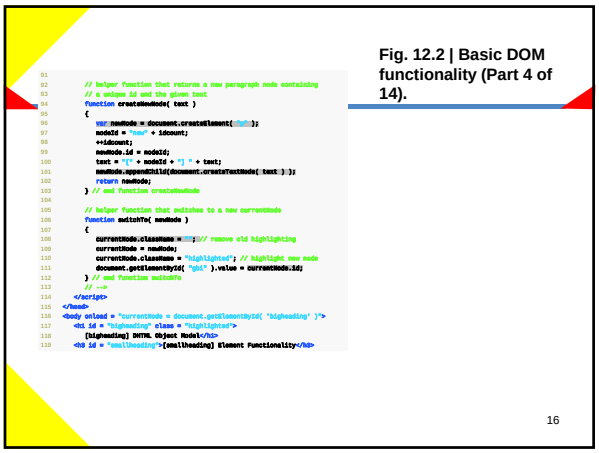
12

Fig. 12.2 | Basic DOM functionality (Part 1 of 14).

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 12.2: dom.html -->
<!-- Basic DOM Functionality. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Basic DOM Functionality</title>
      <style type = "text/css">
         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 }
         .submit      { width: 120px }
      </style>
      <script type = "text/javascript">
         <!--
         var currentNode; // stores the currently highlighted node
         var idcount = 0; // used to assign a unique id to new elements

         // get and highlight an element by its id attribute
         function byId()
         {
            var id = document.getElementById( "gbl" ).value;
            var target = document.getElementById( id );

            if ( target )
               switchTo( target );
         } // end function byId

         // insert a paragraph element before the current element
         // using the insertBefore method
         function insert()
         {
            var newNode = createNewNode(
               document.getElementById( "ins" ).value );
            currentNode.parentNode.insertBefore( newNode, currentNode );
            switchTo( newNode );
         } // end function insert

         // append a paragraph node as the child of the current node
         function appendNode()
         {
            var newNode = createNewNode(
               document.getElementById( "append" ).value );
            currentNode.appendChild( newNode );
            switchTo( newNode );
         } // end function appendNode

         // replace the currently selected node with a paragraph node
         function replaceCurrent()
         {
            var newNode = createNewNode(
               document.getElementById( "replace" ).value );
            currentNode.parentNode.replaceChild( newNode, currentNode );
            switchTo( newNode )
         } // end function replaceCurrent

         // removes the current node
         function remove()
         {
            if ( currentNode.parentNode == document.body )
               alert( "Can't remove a top-level element." );
            else
            {
               var oldNode = currentNode;
               switchTo( oldNode.parentNode );
               currentNode.removeChild( oldNode );
            }
         } // end function remove

         // get and highlight the parent of the current node
         function parent()
         {
            var target = currentNode.parentNode;

            if ( target != document.body )
               switchTo( target );
            else
               alert( "No parent." );
         } // end function parent
```


Fig. 12.2 | Basic DOM functionality (Part 2 of 14).

Fig. 12.2 | Basic DOM functionality (Part 3 of 14).



```
91
92          // helper function that returns a new paragraph node containing
93          // a unique id and the given text
94          function createNewNode( text )
95          {
96             var newNode = document.createElement( "p" );
97             nodeId = "new" + idcount;
98             ++idcount;
99             newNode.id = nodeId;
100            text = "[" + nodeId + "] " + text;
101            newNode.appendChild( document.createTextNode( text ) );
102            return newNode;
103         } // end function createNewNode
104
105         // helper function that switches to a new currentNode
106         function switchTo( newNode )
107         {
108            currentNode.className = ""; // remove old highlighting
109            currentNode = newNode;
110            currentNode.className = "highlighted"; // highlight new node
111            document.getElementById( "qhi" ).value = currentNode.id;
112         } // end function switchTo
113         // -->
114      </script>
115   </head>
116   <body onload = "currentNode = document.getElementById( 'bigheading' )">
117      <h1 id = "bigheading" class = "highlighted">
118         [bigheading] DHTML Object Model</h1>
119      <h2 id = "smallheading">[smallheading] Element Functionality</h2>
```

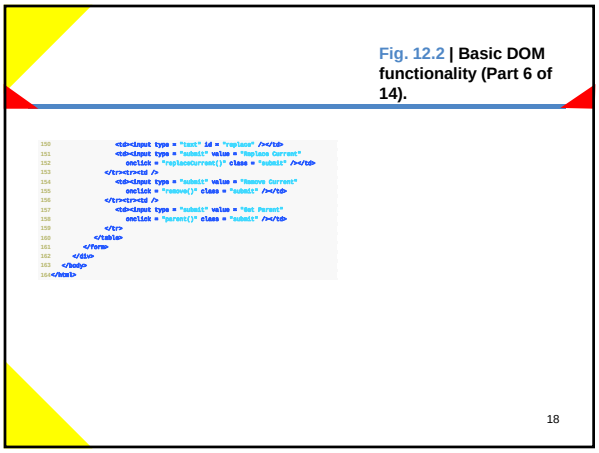
Fig. 12.2 | Basic DOM functionality (Part 4 of 14).

16

Fig. 12.2 | Basic DOM functionality (Part 5 of 14).

```
120      <p id = "para1">[para1] The Document Object Model (DOM) allows for
121         quick, dynamic access to all elements in an XHTML document for
122         manipulation with JavaScript.</p>
123      <p id = "para2">[para2] For more information, check out the
124         "JavaScript and the DOM" section of Deitel's
125         <a id = "link" href = "http://www.deitel.com/javascript">
126            [link] JavaScript Resource Center.</a></p>
127      <p id = "para3">[para3] The buttons below demonstrate:(list)</p>
128      <ul id = "list">
129         <li id = "item1">[item1] getElementById and parentNode</li>
130         <li id = "item2">[item2] insertBefore and appendChild</li>
131         <li id = "item3">[item3] replaceChild and removeChild </li>
132      </ul>
133      <div id = "new" class = "new">
134         <form onsubmit = "return false" action = "">
135            <table>
136               <tr>
137                  <td><input type = "text" id = "qhi"
138                     value = "bigheading" /></td>
139                  <td><input type = "submit" value = "Get By id"
140                     onclick = "byId()" class = "submit" /></td>
141               </tr><tr>
142                  <td><input type = "text" id = "ins" /></td>
143                  <td><input type = "submit" value = "Insert Before"
144                     onclick = "insert()" class = "submit" /></td>
145               </tr><tr>
146                  <td><input type = "text" id = "append" /></td>
147                  <td><input type = "submit" value = "Append Child"
148                     onclick = "appendNode()" class = "submit" /></td>
149               </tr><tr>
```

17



```
150               <td><input type = "text" id = "replace" /></td>
151               <td><input type = "submit" value = "Replace Current"
152                  onclick = "replaceCurrent()" class = "submit" /></td>
153            </tr><tr><td />
154               <td><input type = "submit" value = "Remove Current"
155                  onclick = "remove()" class = "submit" /></td>
156            </tr><tr><td />
157               <td><input type = "submit" value = "Get Parent"
158                  onclick = "parent()" class = "submit" /></td>
159            </tr>
160         </table>
161      </form>
162   </div>
163</body>
164</html>
```

Fig. 12.2 | Basic DOM functionality (Part 6 of 14).

18

Fig. 12.2 | Basic DOM functionality (Part 7 of 14).

19

Fig. 12.2 | Basic DOM functionality (Part 8 of 14).

20

Fig. 12.2 | Basic DOM functionality (Part 9 of 14).

21

Fig. 12.2 | Basic DOM functionality (Part 10 of 14).

22

Fig. 12.2 | Basic DOM functionality (Part 11 of 14).

23

Fig. 12.2 | Basic DOM functionality (Part 12 of 14).

24

Fig. 12.2 | Basic DOM functionality (Part 13 of 14).

25

Fig. 12.2 | Basic DOM functionality (Part 14 of 14).

26

Fig. 12.3 | Using the `links` collection (Part 1 of 3).

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 12.3: collections.html -->
<!-- Using the links collection. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Using Links Collection</title>
      <style type = "text/css">
         body              { font-family: arial, helvetica, sans-serif }
         h1                { font-family: tahoma, geneva, sans-serif;
                             text-align: center }
         p                 { margin: 0px }
         p a               { color: #ff0000 }
         .links            { font-size: 14pt;
                             text-align: justify;
                             margin-left: 10%;
                             margin-right: 10% }
         .link a           { text-decoration: none }
         .link a:hover     { text-decoration: underline }
      </style>
      <script type = "text/javascript">
         <!--
         function processLinks()
         {
            var linksList = document.links; // get the document's links
            var contents = "Links in this page:\n<br />"; 

            // concatenate each link to contents
            for ( var i = 0; i < linksList.length; i++ )
```

27

Fig. 12.3 | Using the `links` collection (Part 2 of 3).

```
         {
            var currentLink = links[ i ];
            contents += "<span class = 'link'>" +
               currentLink.innerHTML.link( currentLink.href ) +
               "</span> | ";
         } // end for

         document.getElementById( "links" ).innerHTML = contents;
      } // end function processLinks
      // -->
   </script>
</head>
<body onload = "processLinks()">
   <h1>Deitel Resource Centers</h1>
   <p><a href = "http://www.deitel.com/">Deitel's website</a> contains
      a rapidly growing
      <a href = "http://www.deitel.com/ResourceCenters.html">list of
      Resource Centers</a> on a wide range of topics. Many Resource
      Centers related to topics covered in this book,
      <a href = "http://www.deitel.com/books/">Internet and World Wide
      Web How to Program, 4th Edition</a>. We have Resource Centers on
      <a href = "http://www.deitel.com/Web20/">Web 2.0</a>,
      <a href = "http://www.deitel.com/Firefox/">Firefox</a> and
      <a href = "http://www.deitel.com/IE7/">Internet Explorer 7</a>,
      <a href = "http://www.deitel.com/XHTML/">XHTML</a>, and
      <a href = "http://www.deitel.com/JavaScript/">JavaScript</a>.
      Watch the list of Deitel Resource Centers for related new
      Resource Centers.</p>
   <div id = "links" class = "links"></div>
</body>
</html>
```

28

Fig. 12.3 | Using the `links` collection (Part 3 of 3).

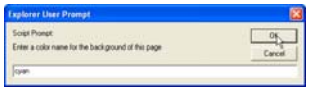
29

Fig. 12.4 | Dynamic styles (Part 1 of 2).

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 12.4: dynamicstyle.html -->
<!-- Dynamic styles. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Dynamic Styles</title>
      <script type = "text/javascript">
         <!--
         function start()
         {
            var inputColor = prompt( "Enter a color name for the text",
               "blue" );
            document.body.style.backgroundColor = inputColor;
         } // end function start
         // -->
      </script>
   </head>
   <body id = "body" onload = "start()">
      <p>Welcome to our website!</p>
   </body>
</html>
```

30

Fig. 12.4 | Dynamic styles (Part 2 of 2).



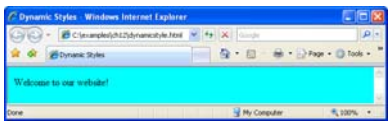


Fig. 12.5 | Dynamic styles used for animation (Part 1 of 7).

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 12.5: coverviewer.html -->
<!-- Dynamic styles used for animation. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Deitel Book Cover Viewer</title>
      <style type = "text/css">
         .thumbs   { width: 100px;
                     height: 375px;
                     padding: 5px;
                     float: left }
         .mainimg  { width: 200px;
                     padding: 5px;
                     float: left }
         .imgCover { height: 375px }
         img       { border: 1px solid black }
      </style>
      <script type = "text/javascript">
         <!--
         var interval = null; // keeps track of the interval
         var speed = 5; // determines the speed of the animation
         var count = 0; // size of the image during the animation

         // called repeatedly to animate the book cover
         function run()
         {
            count += speed;
```


Fig. 12.5 | Dynamic styles used for animation (Part 2 of 7).

```
            // stop the animation when the image is large enough
            if ( count >= 375 )
            {
               window.clearInterval( interval );
               interval = null;
            } // end if

            var bigImage = document.getElementById( "imgCover" );
            bigImage.style.width = .7555 * count + "px";
            bigImage.style.height = count + "px";
         } // end function run

         // inserts the proper image into the main image area and
         // begins the animation
         function display( imgfile )
         {
            if ( interval )
               return;

            var bigImage = document.getElementById( "imgCover" );
            var newNode = document.createElement( "img" );
            newNode.id = "imgCover";
            newNode.src = "thumbs/" + imgfile;
            newNode.alt = "Large Image";
            newNode.className = "imgCover";
            newNode.style.width = "0px";
            newNode.style.height = "0px";
            bigImage.parentNode.replaceChild( newNode, bigImage );
            count = 0; // start the image at size 0
```

Fig. 12.5 | Dynamic styles used for animation (Part 3 of 7).

```
61          interval = window.setInterval( changeImg, 3000 ); ///instance
62       } // end function display
63       // -->
64    </script>
65 </head>
66 <body>
67    <div id = "mainimg" class = "mainimg">
68       <img id = "imgCover" src = "fullsize/1wHhtp4.jpg"
69          alt = "Full cover image" class = "imgCover" />
70    </div>
71    <div id = "thumbs" class = "thumbs" >
72       <img src = "thumbs/1wHhtp4.jpg" alt = "1wHhtp4"
73          onclick = "display( '1wHhtp4.jpg' )" />
74       <img src = "thumbs/chtp6.jpg" alt = "chtp6"
75          onclick = "display( 'chtp6.jpg' )" />
76       <img src = "thumbs/cpphtp6.jpg" alt = "cpphtp6"
77          onclick = "display( 'cpphtp6.jpg' )" />
78       <img src = "thumbs/jhtp7.jpg" alt = "jhtp7"
79          onclick = "display( 'jhtp7.jpg' )" />
80       <img src = "thumbs/vbhtp8.jpg" alt = "vbhtp8"
81          onclick = "display( 'vbhtp8.jpg' )" />
82       <img src = "thumbs/vcsharphtp3.jpg" alt = "vcsharphtp3"
83          onclick = "display( 'vcsharphtp3.jpg' )" />
84    </div>
85 </body>
86 </html>
```


Fig. 12.5 | Dynamic styles used for animation (Part 4 of 7).

Fig. 12.5 | Dynamic styles used for animation (Part 5 of 7).

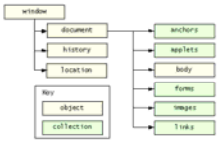
Fig. 12.5 | Dynamic styles used for animation (Part 6 of 7).

37

Fig. 12.5 | Dynamic styles used for animation (Part 7 of 7).

38

Fig. 12.6 | W3C Document Object Model.



39

Fig. 12.7 | Objects and collections in the W3C Document Object Model (Part 1 of 2).

Object or collection	Description
Objects	
`window`	Represents the browser window and provides access to the `document` object contained in the `window`. Also contains `history` and `location` objects.
`document`	Represents the XHTML document rendered in a window. The `document` object provides access to every element in the XHTML document and allows dynamic modification of the XHTML document. Contains several collections for accessing all elements of a given type.
`body`	Provides access to the `body` element of an XHTML document.
`history`	Keeps track of the sites visited by the browser user. The object provides a script programmer with the ability to move forward and backward through the visited sites.
`location`	Contains the URL of the rendered document. When this object is set to a new URL, the browser immediately navigates to the new location.

Fig. 12.7 | Objects and collections in the W3C Document Object Model (Part 2 of 2).

Object or collection	Description
Collections	
`anchors`	Collection contains all the anchor elements (`a`) that have a `name` or `id` attribute. The elements appear in the collection in the order in which they were defined in the XHTML document.
`forms`	Contains all the `form` elements in the XHTML document. The elements appear in the collection in the order in which they were defined in the XHTML document.
`images`	Contains all the `img` elements in the XHTML document. The elements appear in the collection in the order in which they were defined in the XHTML document.
`links`	Contains all the anchor elements (`a`) with an `href` property. The elements appear in the collection in the order in which they were defined in the XHTML document.

Internet & World Wide Web: How to Program by Deitel and Deitel

Document Object Model (DOM): Objects and Collections Chapter 12