

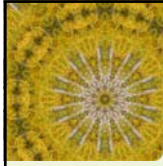
Tutorial 7

Working with Dynamic
Content and Styles



Objectives

- Learn how to create dynamic content under the IE DOM
- Understand the methods and properties of nodes and the node tree
- Learn to create element and text nodes
- Understand how to attach nodes to a Web page document



Objectives

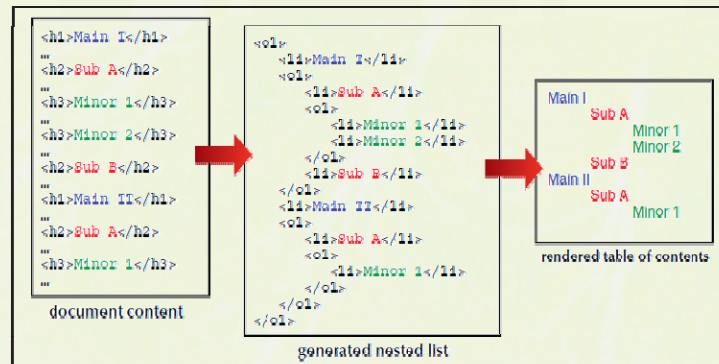
- Apply node properties and styles to create dynamic content
- Work with the properties and methods of attribute nodes
- Work with element attributes



Objectives

- Hide and redisplay Web page objects
- Understand how to create recursive functions to navigate a node tree
- Learn to work with the properties and methods of style sheet objects

Introducing Dynamic Content



Introducing Dynamic Content

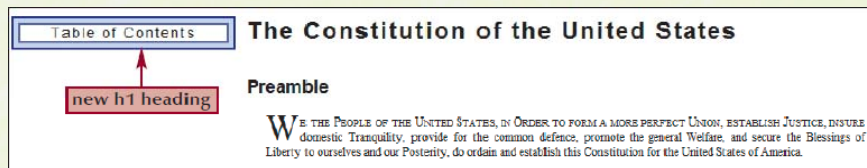
- Inserting HTML Content into an Element
 - Generating a table of contents involves working with **dynamic content**, which is content determined by the operation of a script running within the browser
 - One property that can be used to write content in an element is the innerHTML property
`object.innerHTML = content`



Introducing Dynamic Content

- Inserting HTML Content into an Element

```
function makeTOC() {  
    var TOC = document.getElementById("toc");  
    TOC.innerHTML = "<h1>Table of Contents</h1>";  
}
```



Introducing Dynamic Content

- Dynamic Content in Internet Explorer
 - The innerHTML property is not part of the official specifications for the W3C document object model
 - However, since it has proven valuable and easy to use, it is supported by all browsers
 - If you want to change both the content and the HTML element itself, you use the outerHTML property
`object.outerHTML = content;`



Working with Nodes

- A **node** represents an object within the Web page and Web browser
- The text within an HTML tag can also be treated as a node. For example, the tag

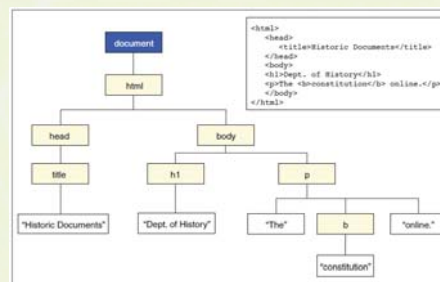
```
<h1>Table of Contents</h1>
```

consists of two nodes: one node for the h1 element and one node for the text string, Table of Contents, contained within that element



Working with Nodes

- The Node Tree
 - Nodes are arranged into a hierarchal structure called a **node tree**, which indicates the relationship between each of the nodes





Working with Nodes

- The Node Tree
 - The parent of all nodes within a document is the **root node**

Expression	Description
<code>node.firstChild</code>	Returns the first child of <i>node</i>
<code>node.lastChild</code>	Returns the last child of <i>node</i>
<code>node.childNodes</code>	Returns a collection containing the children of <i>node</i>
<code>node.previousSibling</code>	Returns the sibling prior to <i>node</i>
<code>node.nextSibling</code>	Returns the sibling after <i>node</i>
<code>node.ownerDocument</code>	Returns the root node of the document
<code>node.parentNode</code>	Returns the parent of <i>node</i>



Working with Nodes

- Node types, names, and values

Node	<code>.nodeType</code>	<code>.nodeName</code>	<code>.nodeValue</code>
Element	1	<i>ELEMENT NAME</i>	null
Attribute	2	<i>attribute name</i>	<i>attribute value</i>
Text	3	<i>#text</i>	<i>text string</i>
Comment	8	<i>#comment</i>	<i>comment text</i>
Document	9	<i>#document</i>	null



Working with Nodes

- Node types, names, and values

Node	.nodeType	.nodeName	.nodeValue
Document	9	#document	null
html	1	HTML	null
head	1	HEAD	null
body	1	BODY	null
title	1	TITLE	null
"Historic Documents"	3	#text	Historic Documents
h1	1	H1	null
"Dept. of History"	3	#text	Dept. of History
p	1	P	null
"The "	3	#text	The
b	1	B	null
"constitution"	3	#text	constitution
"online"	3	#text	online



Working with Nodes

- Creating and Attaching Nodes

Method	Description
<code>document.createAttribute(att)</code>	Creates an attribute node with the name <i>att</i> .
<code>document.createComment(text)</code>	Creates a comment node containing the comment text string <i>text</i> .
<code>document.createElement(elem)</code>	Creates an element node with the name <i>elem</i> .
<code>document.createTextNode(text)</code>	Creates a text node containing the text string <i>text</i> .
<code>node.cloneNode(deep)</code>	Creates a copy of <i>node</i> . If the Boolean parameter <i>deep</i> is true, the copy extends to all descendants of the node object; otherwise, only <i>node</i> is copied.



Working with Nodes

- Creating and Attaching Nodes
 - Unattached nodes and node trees are known as **document fragments** and exist only in a browser's memory

Method	Description
<code>node.appendChild(new)</code>	Appends a <i>new</i> child node to <i>node</i> , attaching it as the last child node
<code>node.insertBefore(new, child)</code>	Inserts a <i>new</i> child node into <i>node</i> , placing it before the <i>child</i> node; if no <i>child</i> is specified the <i>new</i> child node is added as the last child node
<code>node.normalized()</code>	Traverses all child nodes of <i>node</i> ; any adjacent text nodes are merged into a single text node
<code>node.removeChild(old)</code>	Removes the child node <i>old</i> from <i>node</i>
<code>node.replaceChild(new, old)</code>	Replaces the child node <i>old</i> with the child node <i>new</i>



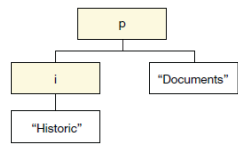
15

*New Perspectives on
JavaScript and AJAX, 2nd Edition*



Working with Nodes

- Creating and Attaching Nodes

Code	Node Tree
<pre>newP = document.createElement("P") newI = document.createElement("I") text1 = document.createTextNode(" Documents") text2 = document.createTextNode("Historic")</pre>	
<pre>newP.appendChild(text1) newI.appendChild(text2)</pre>	
<pre>newP.insertBefore(newI, text1)</pre> <div>Final HTML fragment: <pre><p><i>Historic</i> Documents</p></pre></div>	

16

*New Perspectives on
JavaScript and AJAX, 2nd Edition*



Creating a List of Heading Elements

- Looping Through the Child Node Collection

```
function makeTOC() {  
    var TOC = document.getElementById("toc");  
    TOC.innerHTML = "<h1>Table of Contents</h1>";  
    var TOCList = document.createElement("ol");  
    TOC.appendChild(TOCList);  
}  
  
function createList(object, list) {  
    for (var n = object.firstChild; n != null; n = n.nextSibling) {  
        // loop through all of the nodes within object  
    }  
}
```

creates the list
items for the
table of contents



Creating a List of Heading Elements

- Matching the Heading Elements

```
function createList(object, list) {  
    for (var n = object.firstChild; n != null; n = n.nextSibling) {  
        // loop through all of the nodes within object  
  
        var nodeLevel = levelNum(n);  
        if (nodeLevel != -1) {  
            // node represents a section heading  
        }  
    }  
}
```

locates the section
headings within
the object

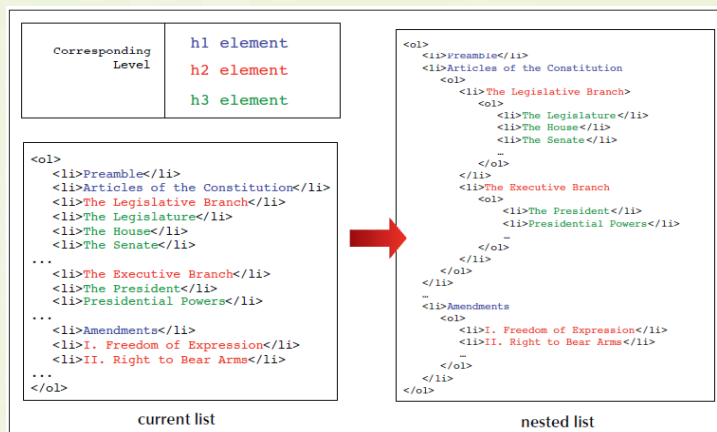
Creating a List of Heading Elements

- Creating the List Item Elements

```
function createList(object, list) {  
  for (var n = object.firstChild; n != null; n = n.nextSibling) {  
    // loop through all of the nodes within object  
  
    var nodeLevel = levelNum(n);  
    if (nodeLevel != -1) {  
      // node represents a section heading  
      // create a list item to match  
      var listItem = document.createElement("li");  
      listItem.innerHTML = n.innerHTML;  
      list.appendChild(listItem);  
    }  
  }  
}
```

text of the list item
comes from the text
of the section heading

Creating a Nested List



Creating a Nested List

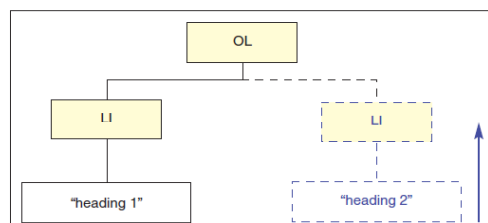
```
function createList(object, list) {  
    var prevLevel = 0; // level of the previous TOC entry  
    for (var n = object.firstChild; n!=null; n = n.nextSibling) {  
        // loop through all of the nodes within object  
        var nodeLevel = levelNum(n);  
        if (nodeLevel != -1) {  
            // node represents a section heading  
            // create a list item to match  
            var listItem = document.createElement("li");  
            listItem.innerHTML = n.innerHTML;  
            if (nodeLevel == prevLevel) {  
                // append the entry to the current list  
            }  
            else if (nodeLevel > prevLevel) {  
                // append the entry to a new nested list  
            }  
            else if (nodeLevel < prevLevel) {  
                // append the entry to a higher-level list  
            }  
        }  
    }  
}
```

replaces the statement
to append the list item

Creating a Nested List

```
<ol>  
  <li>heading 1</li>  
  <li>heading 2</li>  
</ol>
```

HTML Fragment



corresponding node tree

Creating a Nested List

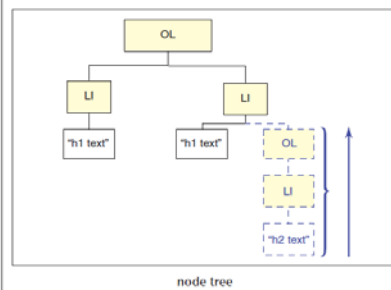
replaces the statement
to append the list item

```
function createlist(object, list) {  
    var prevLevel = 0; // level of the previous TOC entry  
    for (var n = object.firstChild; n != null; n = n.nextSibling) {  
        // loop through all of the nodes within object  
        var nodeLevel = levelNum(n);  
        if (nodeLevel != -1) {  
            // node represents a section heading  
            // create a list item to match  
            var listItem = document.createElement("li");  
            listItem.innerHTML = n.innerHTML;  
            if (nodeLevel == prevLevel) {  
                // append the entry to the current list  
            }  
            else if (nodeLevel > prevLevel) {  
                // append the entry to a new nested list  
            }  
            else if (nodeLevel < prevLevel) {  
                // append the entry to a higher-level list  
            }  
        }  
    }  
}
```

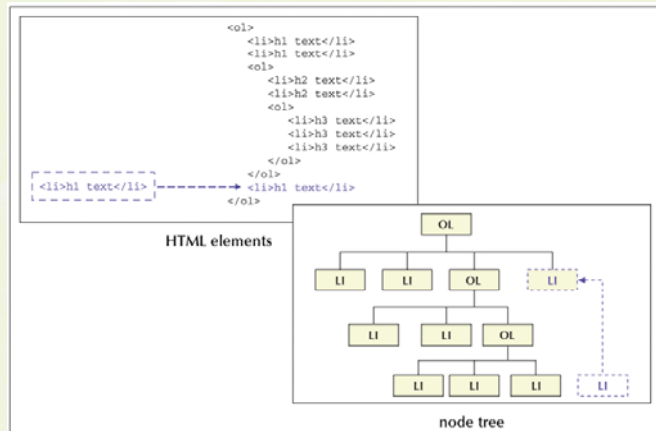
Creating a Nested List

```
<ol>  
  <li>h1 text</li>  
  <li>h1 text  
    <ol>  
      <li>h2 text</li>  
    </ol>  
  </li>  
</ol>
```

HTML Fragment



Creating a Nested List



25

*New Perspectives on
JavaScript and AJAX, 2nd Edition*

Working with Attributes

```

<ol>
  <li><a href = "#head1">Preamble</a></li>
  <li><a href = "#head2">Articles of the Constitution</a></li>
  <ol>
    <li><a href = "#head3">Legislative Branch</a></li>
    <ol>
      <li><a href = "#head4">Section 1: The Legislature</a></li>
      <li><a href = "#head5">Section 2: The House</a></li>
      <li><a href = "#head6">Section 3: The Senate</a></li>
    </ol>
  </ol>
  ...

```

table of contents with links

```

<h1 id = "head1">Preamble</h1>
...
<h1 id = "head2">Articles of the Constitution</h1>
...
<h2 id = "head3">Legislative Branch</h2>
...
<h3 id = "head4">Section 1: The Legislature</h3>
...
<h3 id = "head5">Section 2: The House</h3>
...
<h3 id = "head6">Section 3: The Senate</h3>
....

```

section headings with id values

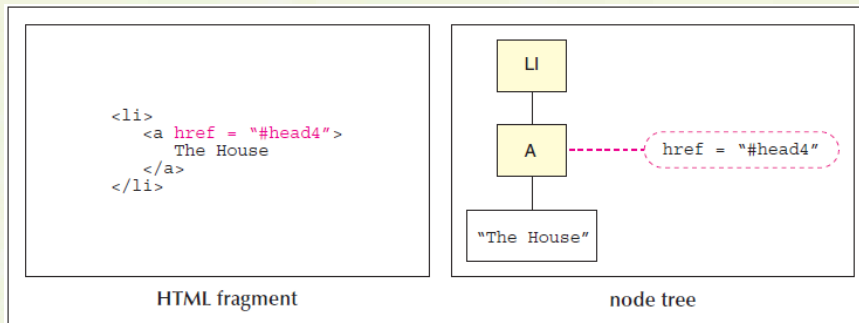
26

*New Perspectives on
JavaScript and AJAX, 2nd Edition*



Working with Attributes

- Attribute Nodes



Working with Attributes

- Attribute Nodes

Method	Description
<code>document.createAttribute(att)</code>	Creates an attribute node with the name <i>att</i>
<code>node.getAttribute(att)</code>	Returns the value of an attribute <i>att</i> from a <i>node</i> to which it has been attached
<code>node.hasAttribute(att)</code>	Returns a Boolean value indicating whether <i>node</i> has the attribute <i>att</i>
<code>node.removeAttribute(att)</code>	Removes the attribute <i>att</i> from <i>node</i>
<code>node.removeAttributeNode(att)</code>	Removes an attribute node <i>att</i> from <i>node</i>
<code>node.setAttribute(att, value)</code>	Creates or changes the <i>value</i> of the attribute <i>att</i> of <i>node</i>



Working with Attributes

- Attributes as Object Properties
 - The document object model also supports a shorthand way of applying attributes as properties of an object
`object.attr = value;`
 - To test whether the listItem node has an id attribute, you can use the following expression:
`listItem.hasAttribute("id")`



Working with Attributes

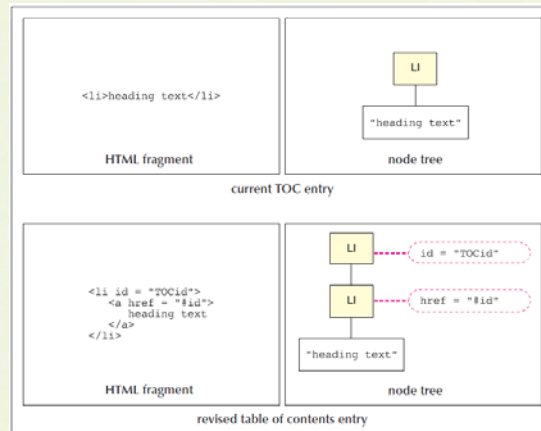
- Setting the Section Heading IDs

adds IDs to section headings that don't already have one

```
function createList(object, list) {  
    var prevLevel = 0; // level of the previous TOC entry  
    var headNum = 0; // running count of section headings  
    for (var n = object.firstChild; n != null; n = n.nextSibling) {  
        // loop through all of the nodes within object  
        var nodeLevel = levelNum(n);  
        if (nodeLevel != -1) {  
            // node represents a section heading  
            // Insert id for the section heading if necessary  
            headNum++;  
            if (n.id == "") {n.id = "head" + headNum;}  
            // create a list item to match  
            var listItem = document.createElement("li");  
            listItem.innerHTML = n.innerHTML;  
            list.appendChild(listItem);  
        }  
    }  
}
```


Working with Attributes

- Inserting Links



31

*New Perspectives on
JavaScript and AJAX, 2nd Edition*

Working with Attributes

- Inserting Links

delete the line to place the heading content into the list item

creates the hypertext link

appends the link to the list item

```
var nodeLevel = levelNum(n);
if (nodeLevel != -1) {
  // node represents a section heading
  // insert id for the section heading if necessary
  headNum++;
  if (n.id == "") {n.id = "head" + headNum;}

  // create a list item to match
  var listItem = document.createElement("li");
  listItem.id = "TOC" + n.id;

  // Create a hypertext link to the section heading
  var linkItem = document.createElement("a");
  linkItem.innerHTML = n.innerHTML;
  linkItem.href = "#" + n.id;

  // Append the hypertext link to the list entry
  listItem.appendChild(linkItem);

  if (nodeLevel == prevLevel) {
    // append the entry to the current list
    list.appendChild(listItem);
  }
}
```

32

*New Perspectives on
JavaScript and AJAX, 2nd Edition*

Expanding and Collapsing a Document

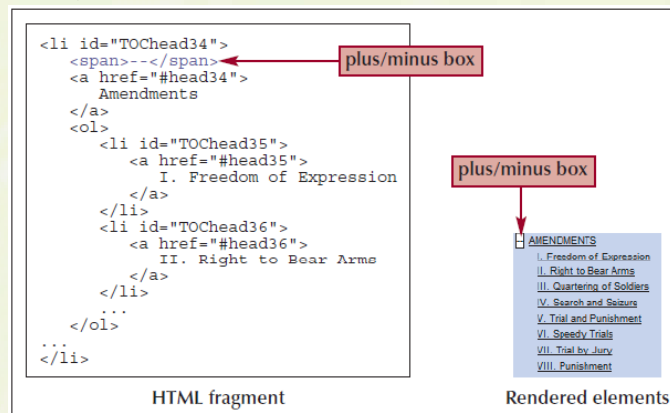


33

*New Perspectives on
JavaScript and AJAX, 2nd Edition*

Expanding and Collapsing a Document

- Creating a plus/minus Box



34

*New Perspectives on
JavaScript and AJAX, 2nd Edition*



Expanding and Collapsing a Document

- Creating a plus/minus Box

places a plus/minus box before each nested list

```
else if (nodeLevel > prevLevel) {
    // append the entry to a new nested list
    var nestedList = document.createElement("ol");
    nestedList.appendChild(listItem);

    list.lastChild.appendChild(nestedList);

    // Add plus/minus box before the text of the nested list
    var plusMinusBox = document.createElement("span");
    plusMinusBox.innerHTML = "--";
    nestedList.parentNode.insertBefore(plusMinusBox, nestedList.previousSibling);

    list = nestedList;
    prevLevel = nodeLevel;
}
```

35

*New Perspectives on
JavaScript and AJAX, 2nd Edition*



Expanding and Collapsing a Document

- Adding Event Handlers to the plus/minus Boxes

runs the expandCollapse() function when the user clicks a plus/minus box

```
else if (nodeLevel > prevLevel) {
    // append the entry to a new nested list
    var nestedList = document.createElement("ol");
    nestedList.appendChild(listItem);

    list.lastChild.appendChild(nestedList);

    // Add plus/minus box before the text of the nested list
    var plusMinusBox = document.createElement("span");
    plusMinusBox.innerHTML = "--";
    addEvent(plusMinusBox, "click", expandCollapse, false);
    nestedList.parentNode.insertBefore(plusMinusBox, nestedList.previousSibling);

    list = nestedList;
    prevLevel = nodeLevel;
}

else if (nodeLevel < prevLevel) {
    // append the entry to a higher level list
    var levelUp = prevLevel - nodeLevel;
    for (var i = 1; i <= levelUp; i++) {list = list.parentNode.parentNode;}

    list.appendChild(listItem);
    prevLevel = nodeLevel;
}
}

function expandCollapse(e) {
    var plusMinusBox = e.target || event.srcElement;

    // toggle the plus and minus symbol
    if (plusMinusBox.innerHTML == "--") plusMinusBox.innerHTML = "+";
    else plusMinusBox.innerHTML = "--";
}
```

36

*New Perspectives on
JavaScript and AJAX, 2nd Edition*



Expanding and Collapsing a Document

- Hiding and Display Objects

```
function expandCollapse(e) {  
    var plusMinusBox = e.target || event.srcElement;  
    var nestedList = plusMinusBox.nextSibling.nextSibling;  
  
    // Toggle the plus and minus symbol  
    if (plusMinusBox.innerHTML == "--") plusMinusBox.innerHTML = "+"  
    else plusMinusBox.innerHTML = "--";  
  
    // Toggle the display style of the nested list  
    if (nestedList.style.display == "none") nestedList.style.display = ""  
    else nestedList.style.display = "none";  
}
```

Table of Contents	
PREAMBLE	
ARTICLES OF THE CONSTITUTION	
Article I - The Legislative Branch	
Article II - The Executive Branch	
Article III - The Judicial Branch	
Section 1: Judicial Powers	
Section 2: Trial by Jury	
Section 3: Treason	
Article IV - The States	
Article V - Amendment Process	
Article VI - The United States	
Article VII - Ratification	
AMENDMENTS	



Expanding and Collapsing a Document

- Expanding and Collapsing the Source Document

```
function expandCollapse(e) {  
    var plusMinusBox = e.target || event.srcElement;  
    var nestedList = plusMinusBox.nextSibling.nextSibling;  
  
    // Toggle the plus and minus symbol  
    if (plusMinusBox.innerHTML == "--") plusMinusBox.innerHTML = "+"  
    else plusMinusBox.innerHTML = "--";  
  
    // Toggle the display style of the nested list  
    if (nestedList.style.display == "none") nestedList.style.display = ""  
    else nestedList.style.display = "none";  
}  
  
function expandCollapseDoc() {  
    var displayStatus = "";  
    for (var n = sourceDoc.firstChild; n != null; n = n.nextSibling) {  
        var nodeLevel = levelNum(n);  
  
        if (nodeLevel != -1) {  
            // determine the display status of the TOC entry  
        }  
  
        if (n.nodeType == 1) { // node represents a page element  
            // apply the current display status to the node  
        }  
    }  
}
```

Expanding and Collapsing a Document

- Expanding and Collapsing the Source Document

```
function expandCollapse(e) {
    var plusMinusBox = e.target || event.srcElement;
    var nestedList = plusMinusBox.nextSibling.nextSibling;

    // Toggle the plus and minus symbol
    if (plusMinusBox.innerHTML == "+") plusMinusBox.innerHTML = "-";
    else plusMinusBox.innerHTML = "+";

    // Toggle the display style of the nested list
    if (nestedList.style.display == "none") nestedList.style.display = "";
    else nestedList.style.display = "none";

    // expand and collapse the source document to match the TOC
    expandCollapseDoc();
}

function expandCollapseDoc() {
    var displayStatus = "";
    for (var n = sourceDoc.firstChild; n != null; n = n.nextSibling) {
        var nodeLevel = levelNum(n);

        if (nodeLevel != -1) {
            // determine the display status of the TOC entry
            var tocEntry = document.getElementById("toc" + n.id);
            if (!isHidden(tocEntry)) displayStatus = "none";
            else displayStatus = "";
        }

        if (n.nodeType == 1) { // node represents a page element
            // apply the current display status to the node
            n.style.display = displayStatus;
        }
    }
}
```

expands and collapses the source document to match the TOC

determines whether the TOC entry is visible

applies the display status to the current node in the source document

39

*New Perspectives on
JavaScript and AJAX, 2nd Edition*

Expanding and Collapsing a Document

- Testing the Dynamic TOC

```
<title>Treaty with the Creek Indians: 1790</title>
<link href="web.css" rel="stylesheet" type="text/css" />
<script src="toc.js" type="text/javascript"></script>
</head>

<body>
  <div id="toc"></div>

  <div id="logo"></div>
  <div id="logosub">Department of History<br />Midwest University</div>
  <div id="docTitle"><h1>Treaty with the Creek Indians: 1790</h1></div>

  <div id="doc">
    <h1>Preamble</h1>
    <p>THE parties being desirous of establishing permanent peace and
    friendship between the United States and the said Creek Nation, and
    the citizens and members thereof, and to remove the causes of war by
    ascertaining their limits, and making other necessary, just and
    friendly arrangements: The President of the United States, by Henry
    Knox, Secretary for the Department of War, whom he hath constituted with
    full powers for these purposes, by and with the advice and consent of
    the Senate of the United States, and the Creek Nation, by the
    undersigned Kings, Chiefs, and Warriors, representing the said Nation,
    have agreed to the following articles. </p>

    </div>
</body>
</html>
```

40

*New Perspectives on
JavaScript and AJAX, 2nd Edition*



Traversing the Node Tree with Recursion

- **Recursion** is a programming technique in which a function calls itself repeatedly until a stopping condition is met

```
function countNodes(node, nodeCount) {  
    for (var n = node.firstChild; n != null; n = n.nextSibling) {  
        nodeCount++;  
        countNodes(n, nodeCount);  
    }  
    return nodeCount;  
}
```

41

*New Perspectives on
JavaScript and AJAX, 2nd Edition*



Switching Between Style Sheets

Historic Documents Online

The Constitution of the United States

Preamble

Articles of the Constitution

Article I - The Legislative Branch

Section 1: The Legislature

Section 2: The House

Section 3: The Senate

Web view

Web view

print view

The Constitution of the United States

Preamble

Articles of the Constitution

Article I - The Legislative Branch

Section 1: The Legislature

Section 2: The House

Section 3: The Senate

large text view

42

*New Perspectives on
JavaScript and AJAX, 2nd Edition*



Switching Between Style Sheets

- Style sheets can be classified as persistent, preferred, and alternate:
 - **Persistent style sheets** are always active
 - **Preferred style sheets** are turned on by default, but can be turned off by actions of the user
 - **Alternate style sheets** are not turned on by default, but can be turned on as an alternate to the preferred style sheet



Switching Between Style Sheets

Historic Documents Online
Department of History
Midwest University

1600 1650 1700 1750 1800 1850 1900 1950 2000 >>

Table of Contents

☐ PREAMBLE
☐ ARTICLES OF THE CONSTITUTION
 ☐ Article I - The Legislative Branch
 Section 1: The Legislature
 Section 2: The House
 Section 3: The Senate
 ☐ Article II - The Executive Branch
 Section 1: The President
 Section 2: Presidential Powers
 Section 3: Electors
 ☐ Article III - The Judicial Branch
 Section 1: The Supreme Court
 Section 2: Lower Courts

The Constitution of the United States

Preamble

WE THE PEOPLE OF THE UNITED STATES, in ORDER to form a more perfect Union, establish Justice, insure domestic Tranquillity, provide for the common defence, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our Posterity, do ordain and establish this Constitution for the United States of America.

Articles of the Constitution

Article I - The Legislative Branch

Section 1: The Legislature
All legislative Powers herein granted shall be vested in a Congress of the United States, which shall consist of a Senate and House of Representatives.

Web View
Print View
Large Text View

style buttons generated from the list of preferred and alternate style sheets in the HTML file



Switching Between Style Sheets

- Code to populate the allStyles array

```
addEvent(window, "load", makeStyleButtons, false);  
var allStyles = new Array();  
function makeStyleButtons() {  
    var allLinks = document.getElementsByTagName("link");  
    // Create an array of preferred or alternate style sheets  
    for (var i = 0; i < allLinks.length; i++) {  
        if ((allLinks[i].rel == "stylesheet" || allLinks[i].rel == "alternate stylesheet")  
            && allLinks[i].title != "") {  
            allStyles.push(allLinks[i]);  
        }  
    }  
}
```

allStyles array contains all link elements created for preferred or alternate style sheets



Switching Between Style Sheets

- Creating a form button for each style sheet

```
// Create an array of preferred or alternate style sheets  
for (var i = 0; i < allLinks.length; i++) {  
    if ((allLinks[i].rel == "stylesheet" || allLinks[i].rel == "alternate stylesheet")  
        && allLinks[i].title != "") {  
        allStyles.push(allLinks[i]);  
    }  
}  
// Create buttons for each preferred or alternate style sheet  
var styleBox = document.createElement("div");  
for (var i = 0; i < allStyles.length; i++) {  
    styleButton = document.createElement("input");  
    styleButton.type = "button";  
    styleButton.value = allStyles[i].title + " view";  
    styleButton.title = allStyles[i].title;  
    styleBox.appendChild(styleButton);  
}
```

box containing the style buttons

creates an input button for each style sheet

appends each button to the style box



Switching Between Style Sheets

- You can disable a style sheet using the command
`styleSheet.disabled = true`
- You can enable it with the command
`styleSheet.disabled = false`
- Code to initialize the style sheets

```
// Create buttons for each preferred or alternate style sheet
var styleBox = document.createElement("div");
for (var i = 0; i < allStyles.length; i++) {
    // Initialize the style sheets
    if (allStyles[i].rel == "stylesheet") {
        allStyles[i].disabled = false;
    } else {
        allStyles[i].disabled = true;
    }
    styleButton = document.createElement("input");
    styleButton.type = "button";
    styleButton.value = allStyles[i].title + " view";
    styleButton.title = allStyles[i].title;
}
```



Switching Between Style Sheets

- Code to switch between styles

```
// Apply an event handler to the style button
styleButton.onclick = changeStyle;
styleBox.appendChild(styleButton);

// Define the styles of the box containing the buttons
styleBox.style.width = "125px";
styleBox.style.cssFloat = "right";
styleBox.style.styleFloat = "right";
styleBox.style.margin = "5px 5px 10px 10px";

// Add the style box to the source document
var sourceDoc = document.getElementsByTagName("doc");
sourceDoc.insertBefore(styleBox, sourceDoc.firstChild);
}

function changeStyle() {
    for (var i = 0; i < allStyles.length; i++) {
        if (allStyles[i].title == this.title) {
            allStyles[i].disabled = false;
        } else {
            allStyles[i].disabled = true;
        }
    }
}
```

runs the `changeStyle()` function whenever a style button is clicked

loops through all of the style sheets and enables only the one corresponding to the style button



Switching Between Style Sheets

- Properties of the style sheet object

Property	Description
<code>styleSheet.cssText</code>	The text of the declarations in the style sheet (IE DOM)
<code>styleSheet.disabled</code>	Returns a Boolean value indicating whether the style sheet has been disabled (true) or has been enabled (false)
<code>styleSheet.href</code>	The url of the style sheet; for embedded style sheets, the href value is an empty text string [read-only]
<code>styleSheet.media</code>	A text string containing the list of media types associated with the style sheet [read-only]
<code>styleSheet.rules</code>	Returns the collection of rules within the style sheet (IE DOM)
<code>styleSheet.cssRules</code>	Returns the collection of rules within the style sheet (W3C DOM)
<code>styleSheet.title</code>	The title of the style sheet [read-only]
<code>styleSheet.type</code>	The MIME type of the style sheet [read-only]