# The DOM, or Web Page As Tree

Bok, Jong Soon
javaexpert@nate.com
https://github.com/swacademy

# Overview

- In 1998, the first release of the W3C DOM Level 1.
- In 2000, followed DOM Level 2.
- In 2004, released DOM Level 3.
- DOM Level 4 is currently being developed.
- Draft version 6 was released in December 2012.

# What is DOM?

- Is an fundamental API for **manipulating** HTML and XML documents.
- *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically* **access** *and* **update** *the content, structure, and style of a document."*
- Refer to https://developer.mozilla.org/en-US/docs/DOM

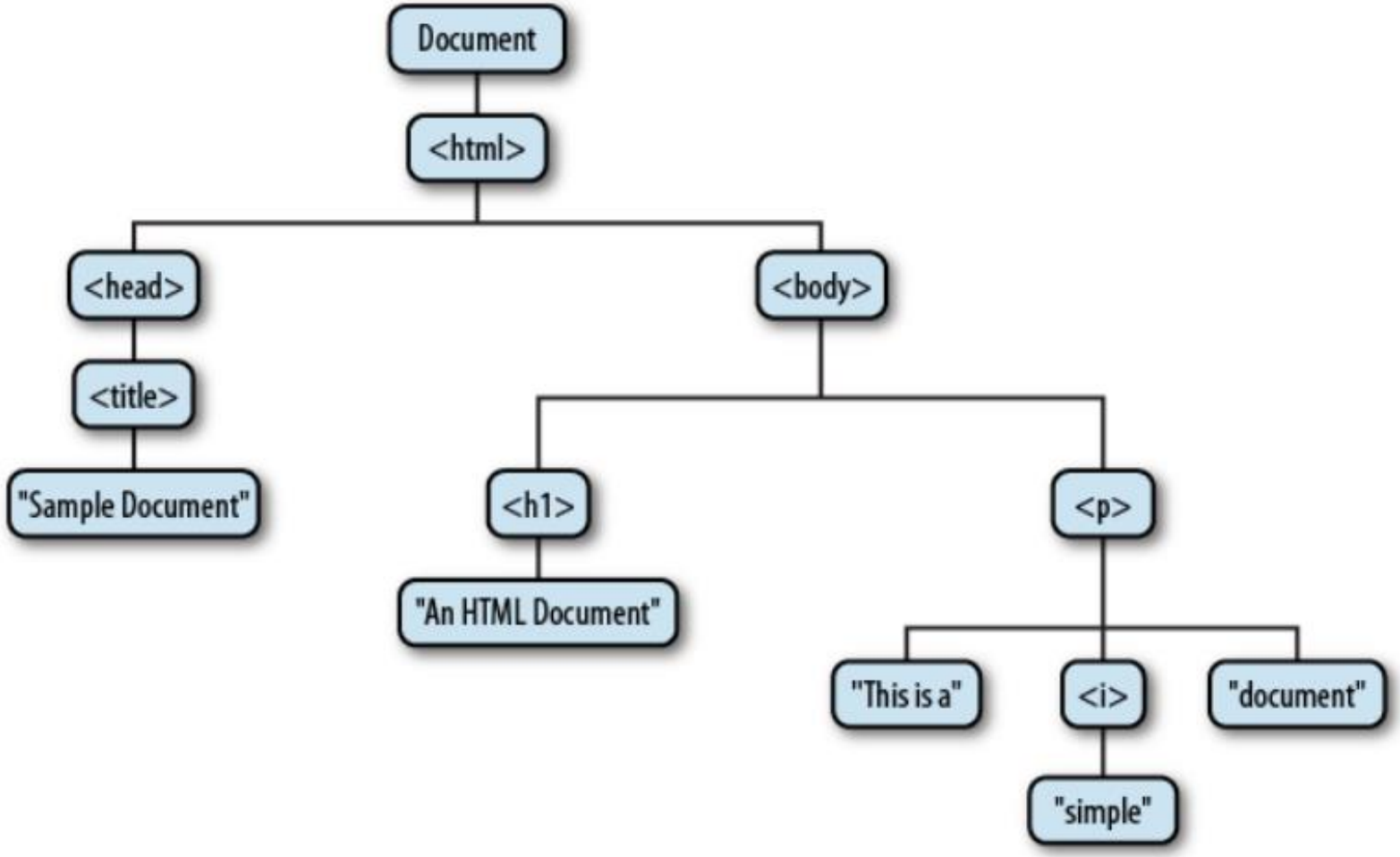
# What is DOM? (Cont.)

- Every Window object has a `document` property that refers to a Document object.
- The nested elements of an HTML or XML document are represented in the DOM as a *tree* of objects.
- The tree representation of an HTML document contains nodes representing HTML tags or elements, such as `<body>` and `<p>`, and nodes representing strings of text.
- An HTML document may also contain nodes representing HTML comments.

# What is DOM? (Cont.)

```
2  <html>
3      <head>
4          <title> Sample Document </title>
5      </head>
6      <body>
7          <h1>An HTML Document</h1>
8          <p>This is a <i>simple</i> document.
9      </body>
10 </html>
```

# What is DOM? (Cont.)



# What is DOM? (Cont.)

- The node directly above a node is the *parent* of that node.
- The nodes one level directly below another node are the *children* of that node.
- Nodes at the same level, and with the same parent, are *siblings*.
- The set of nodes any number of levels below another node are the *descendants* of that node.
- And the parent, grandparent, and all other nodes above a node are the *ancestors* of that node.

# What is DOM? (Cont.)

- Can change all the HTML elements in the page.
- Can change all the HTML attributes in the page.
- Can change all the CSS styles in the page.
- Can remove existing HTML elements and attributes.
- Can add new HTML elements and attributes.
- Can react to all existing HTML events in the page.
- Can create new HTML events in the page.

# What is DOM? (Cont.)

- The W3C DOM standard is separated into 3 different parts:
  - Core DOM - standard model for all document types
  - XML DOM - standard model for XML documents
  - HTML DOM - standard model for HTML documents

# A Tale of Two Interface

- When the W3C released the first version of the DOM, the organization also released two different APIs:
  - The Core API
  - The HTML API.
- A good source for an overview of the different DOM specifications is the OASIS Cover Pages article at: http://xml.coverpages.org/dom.html .

# A Tale of Two Interface (Cont.)
## -The Core API

- The DOM Core is a language, and model-neutral API.
- Can implement in any language, not just JavaScript, and for any XML-based model, not just XHTML.
- Provides all the functionality you need to modify, remove, or create web page content.
- The Core API is a generic API.

# A Tale of Two Interface (Cont.)
## -The Core API

- Consists of objects such as `Node` and `NodeList`, `Attr`, `Element`, and the all-important `Document`.
- Provides a basic set of data types and expected behaviors that browsers must support.

# A Tale of Two Interface (Cont.)
## -The HTML API

- But, Prior to the release of the DOM specification, browsers had already implemented the BOM in various forms, some proprietary and some not.
- To maintain backward compatibility with previous work, the W3C also released a custom subset of the DOM API: the DOM HTML API.

# A Tale of Two Interface (Cont.)
## -The HTML API

- Is an object-oriented, hierarchical view of the web page, with objects mapped to HTML elements.
- `HTMLDocumentElement` for the `document`, `HTMLBodyElement` for the `body`, and so on.

# What is the HTML DOM ?

- Is a standard **object** model and **programming interface** for HTML.
- Defines:
  - The HTML elements as **objects**
  - The **properties** of all HTML elements
  - The **methods** to access all HTML elements
  - The **events** for all HTML elements
- Is a standard for how to get, change, add, or delete HTML elements.

# The DOM HTML APIs

- The DOM (HTML) Level 1 specification
  - http://www.w3.org/TR/REC-DOM-Level-1/
- Associated ECMAScript binding
  - http://www.w3.org/TR/REC-DOM-Level-1/ecma-script-language-binding.html .
- The DOM (HTML) Level 2 specification
  - http://www.w3.org/TR/DOM-Level-2-HTML/
- Associated ECMAScript binding
  - http://www.w3.org/TR/DOM-Level-2-HTML/ecma-script-binding.html .

# The DOM HTML Objects and Their Properties

- Is a set of object *interfaces* rather than actual object classes.
- Methods are *actions* you can perform (on HTML Elements).
- Properties are *values* (of HTML Elements) that you can set or change.

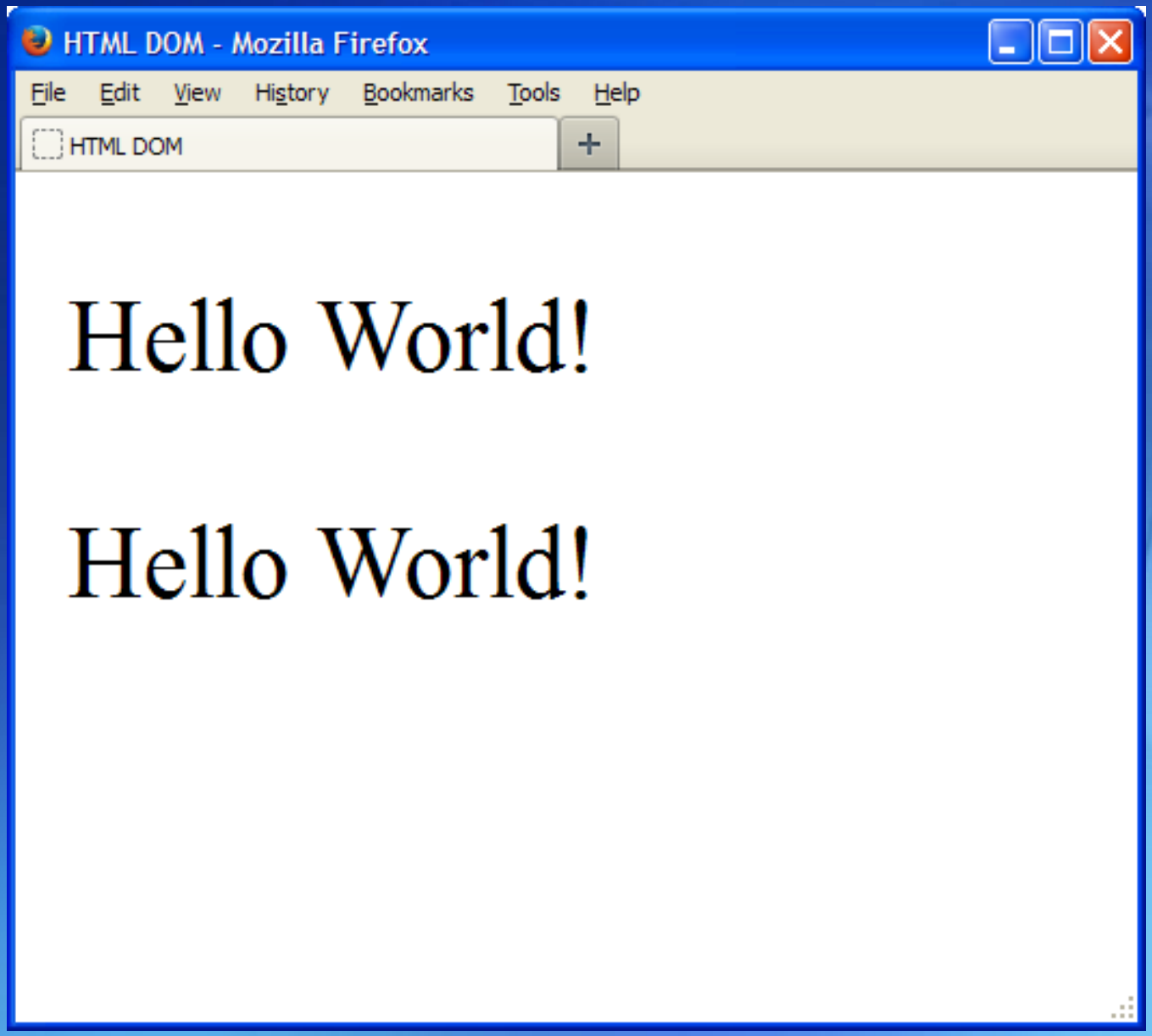
# Lab1 : HTML DOM

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - htmldom.html

# Lab1 : htmldom.html

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title> HTML DOM </title>
5          <meta charset="utf-8">
6      </head>
7      <body>
8          <p id="intro">Hello World!</p>
9
10         <script>
11             txt=document.getElementById("intro").innerHTML;
12             document.write(txt);
13         </script>
14     </body>
15 </html>
```

# Lab1 : Result


HTML DOM - Mozilla Firefox
File Edit View History Bookmarks Tools Help
HTML DOM
Hello World!
Hello World!

# The DOM HTML Objects and Their Properties (Cont.)

- The `getElementById` Method
  - The most common way to access an HTML element is to use the `id` of the element.
- The `innerHTML` Property
  - The easiest way to get the content of an element.
  - Is useful for getting or replacing the content of HTML elements.

# The DOM HTML Objects and Their Properties (Cont.)

- Finding HTML Elements

| Method | Description |
| --- | --- |
| document.getElementById() | Finding an element by element id |
| document.getElementsByTagName() | Finding elements by tag name |
| document.getElementsByClassName() | Finding elements by class name |
| document.forms[] | Finding elements by HTML element objects |

# The DOM HTML Objects and Their Properties (Cont.)

- Changing HTML Elements

| Method | Description |
|---|---|
| document.write(text) | Writing into the HTML output stream |
| document.getElementsById(id).innerHTML = | Changing the inner HTML of an element |
| document.getElementsById(id).attribute = | Changing the attribute of an element |
| document.getElementById(id).style.attribute = | Changing the style of an HTML element |

# The DOM HTML Objects and Their Properties (Cont.)

- Adding and Deleting Elements

| Method | Description |
|---|---|
| document.createElement() | Create an HTML element |
| document.removeChild() | Remove an HTML element |
| document.appendChild() | Add an HTML element |
| document.replaceChild() | replace an HTML element |

- Adding Events Handlers

| Method | Description |
|---|---|
| document.getElementById(id).onclick=function(){code} | Adding event handler code to an onclick event |

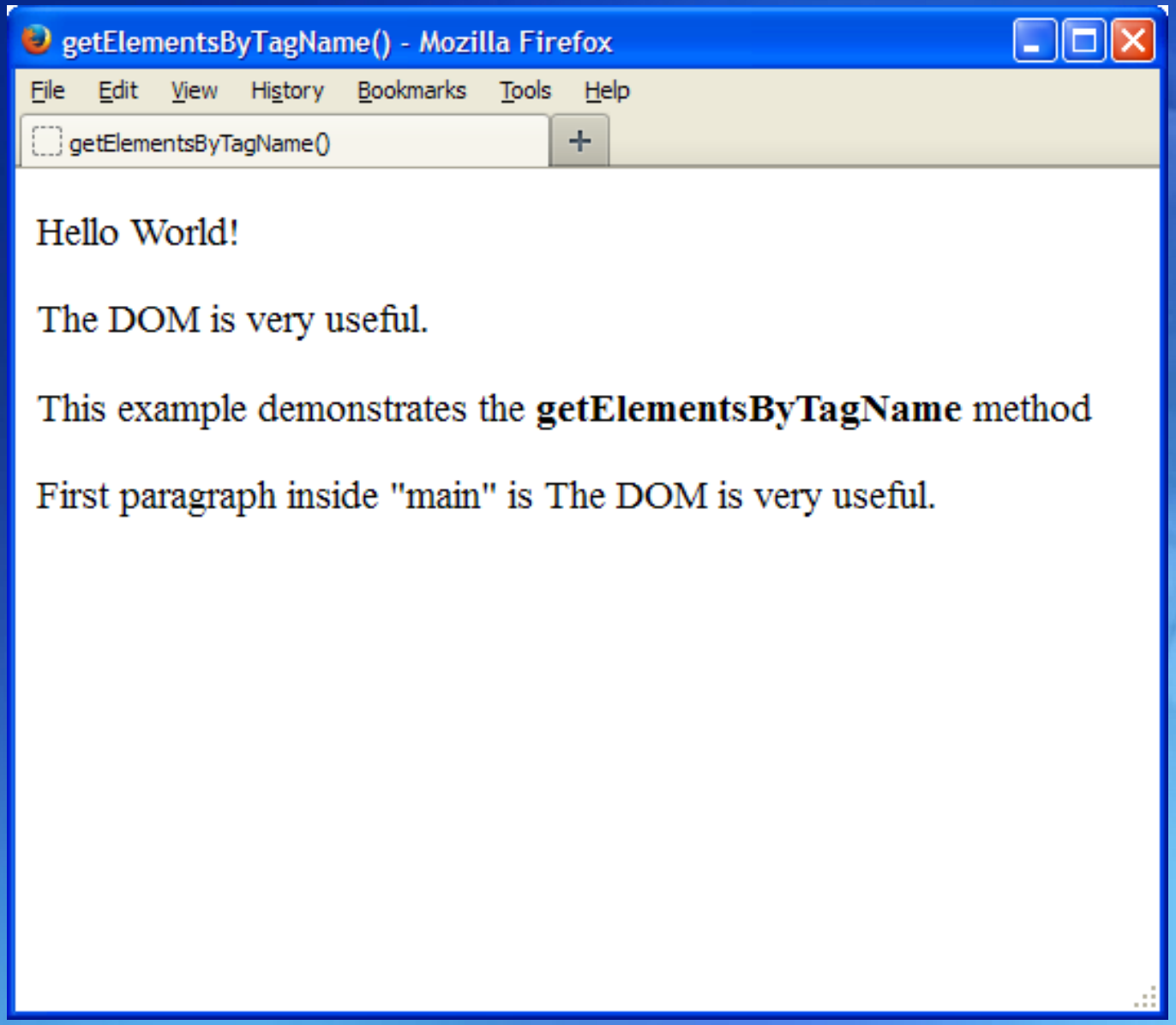
# Lab2 : getElementsByTagName

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - getElementsByTagName.html

# Lab2 : getElementsByTagName.html

```
7   <body>
8     <p>Hello World!</p>
9
10    <div id="main">
11      <p>The DOM is very useful.</p>
12      <p>This example demonstrates the
13        <b>getElementsByTagName</b> method</p>
14    </div>
15
16    <script>
17      var x=document.getElementById("main");
18      var y=x.getElementsByTagName("p");
19      document.write('First paragraph inside "main" is ' + y[0].innerHTML);
20    </script>
21  </body>
```

# Lab2 : Result


getElementsByTagName() - Mozilla Firefox
File Edit View History Bookmarks Tools Help
getElementsByTagName()
Hello World!
The DOM is very useful.
This example demonstrates the getElementsByTagName method
First paragraph inside "main" is The DOM is very useful.

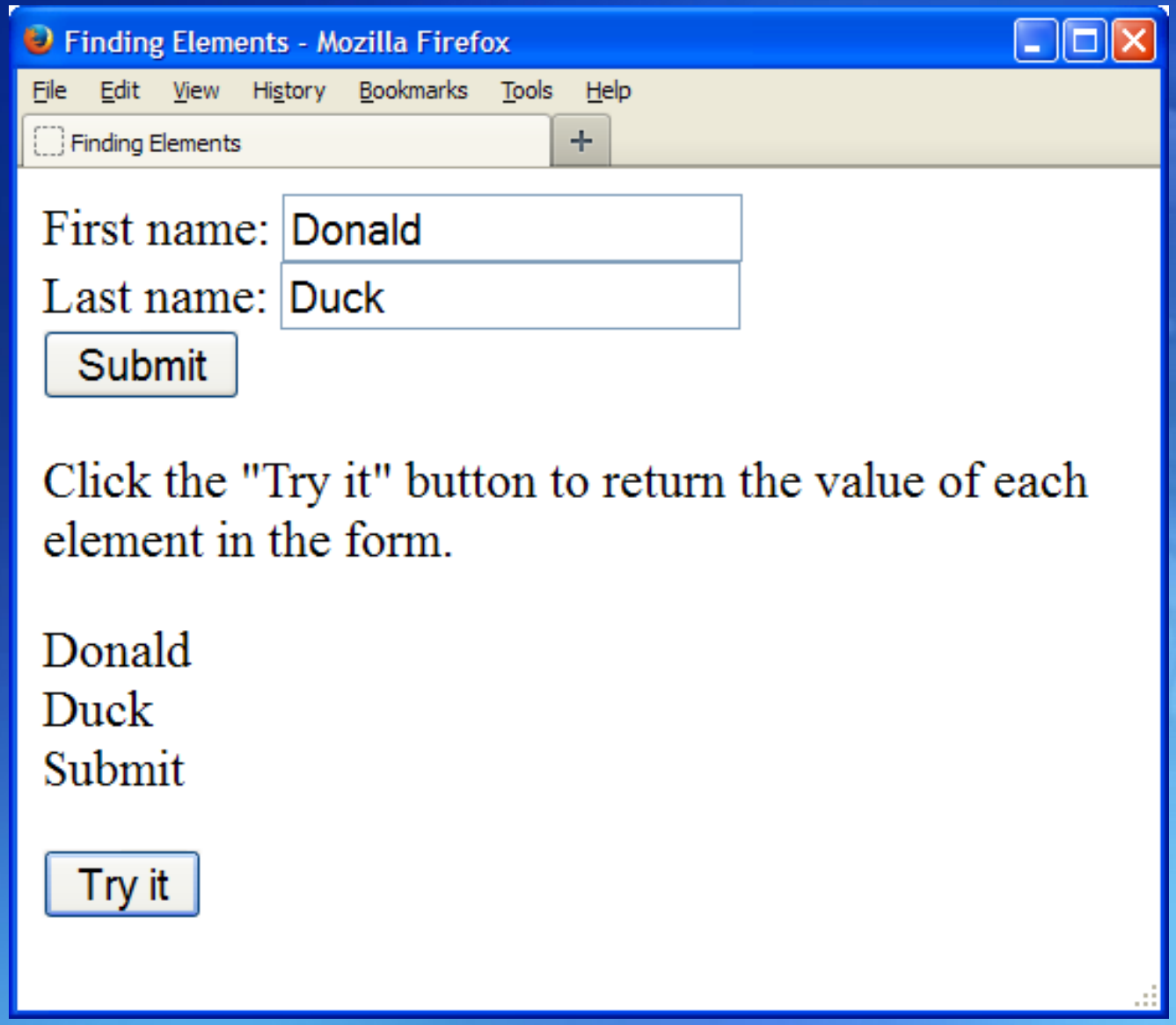
# Lab3 : Finding HTML Elements

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - findelements.html

# Lab3 : findElements.html

```
7   <body>
8     <form id="frm1" action="form_action.asp">
9       First name: <input type="text" name="fname" value="Donald"><br>
10      Last name: <input type="text" name="lname" value="Duck"><br>
11      <input type="submit" value="Submit">
12    </form>
13    <p>Click the "Try it" button to return the value of each element in the form.</p>
14    <p id="demo"></p>
15    <button onclick="myFunction()">Try it</button>
16    <script>
17      function myFunction() {
18        var x = document.getElementById("frm1");
19        var txt = "";
20        for (var i=0;i<x.length;i++){
21          txt = txt + x.elements[i].value + "<br>";
22        }
23        document.getElementById("demo").innerHTML=txt;
24      }
25    </script>
26  </body>
```

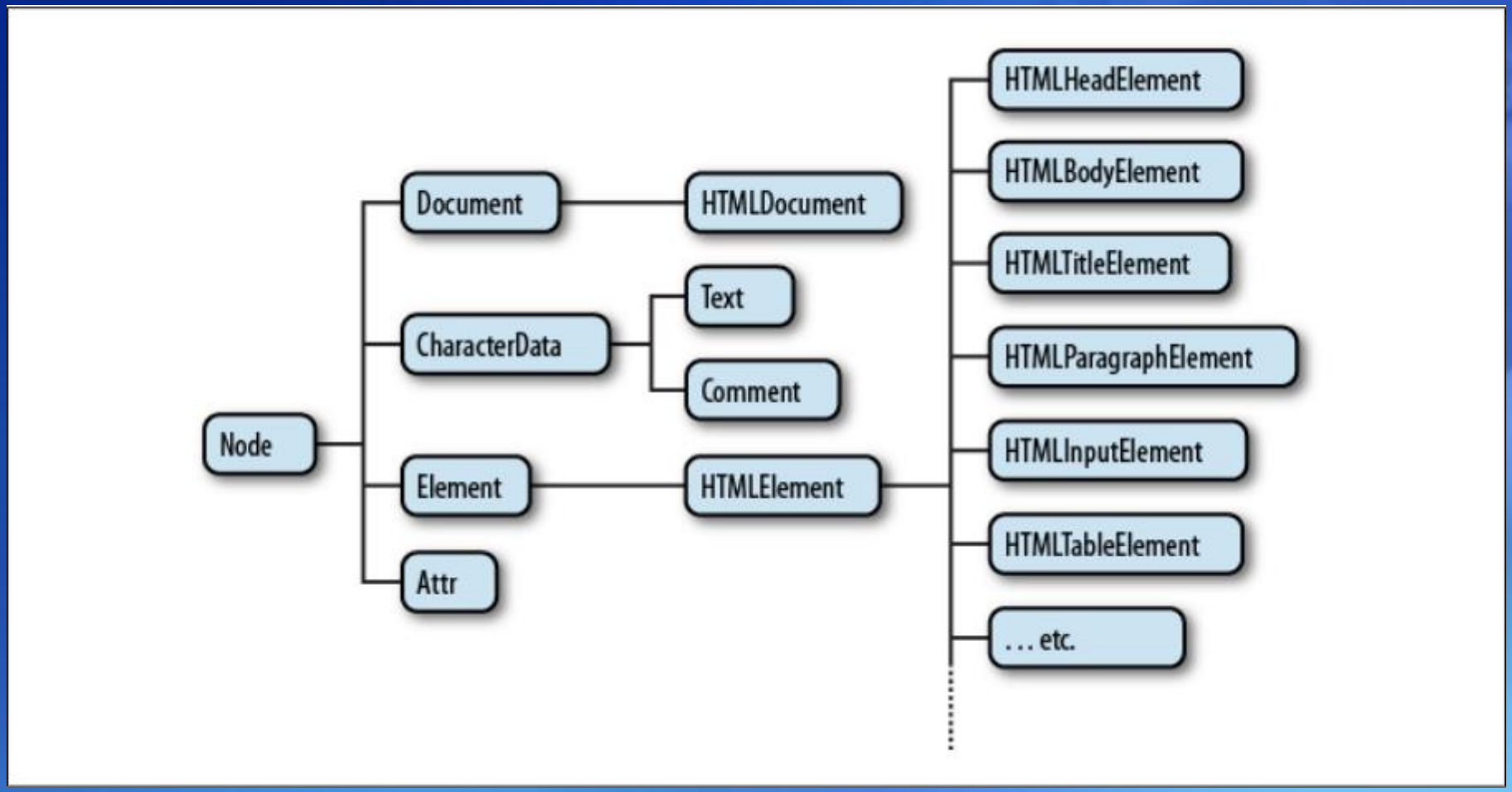
# Lab3 : Result


Finding Elements - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Finding Elements
First name: Donald
Last name: Duck
Submit
Click the "Try it" button to return the value of each element in the form.
Donald
Duck
Submit
Try it

# The DOM HTML Objects and Their Properties (Cont.)

- Most of the DOM (HTML) objects also inherit from `HTMLElement`, which has the following properties, based on the allowable element attributes:
  - `id` : the element identifier
  - `title` : a descriptive title for the element's contents
  - `lang` : the language of the element and its contents
  - `dir` : the directionality of the text (left to right, right to left)
  - `className` : equivalent to the `class` attribute

# The DOM HTML Objects and Their Properties (Cont.)



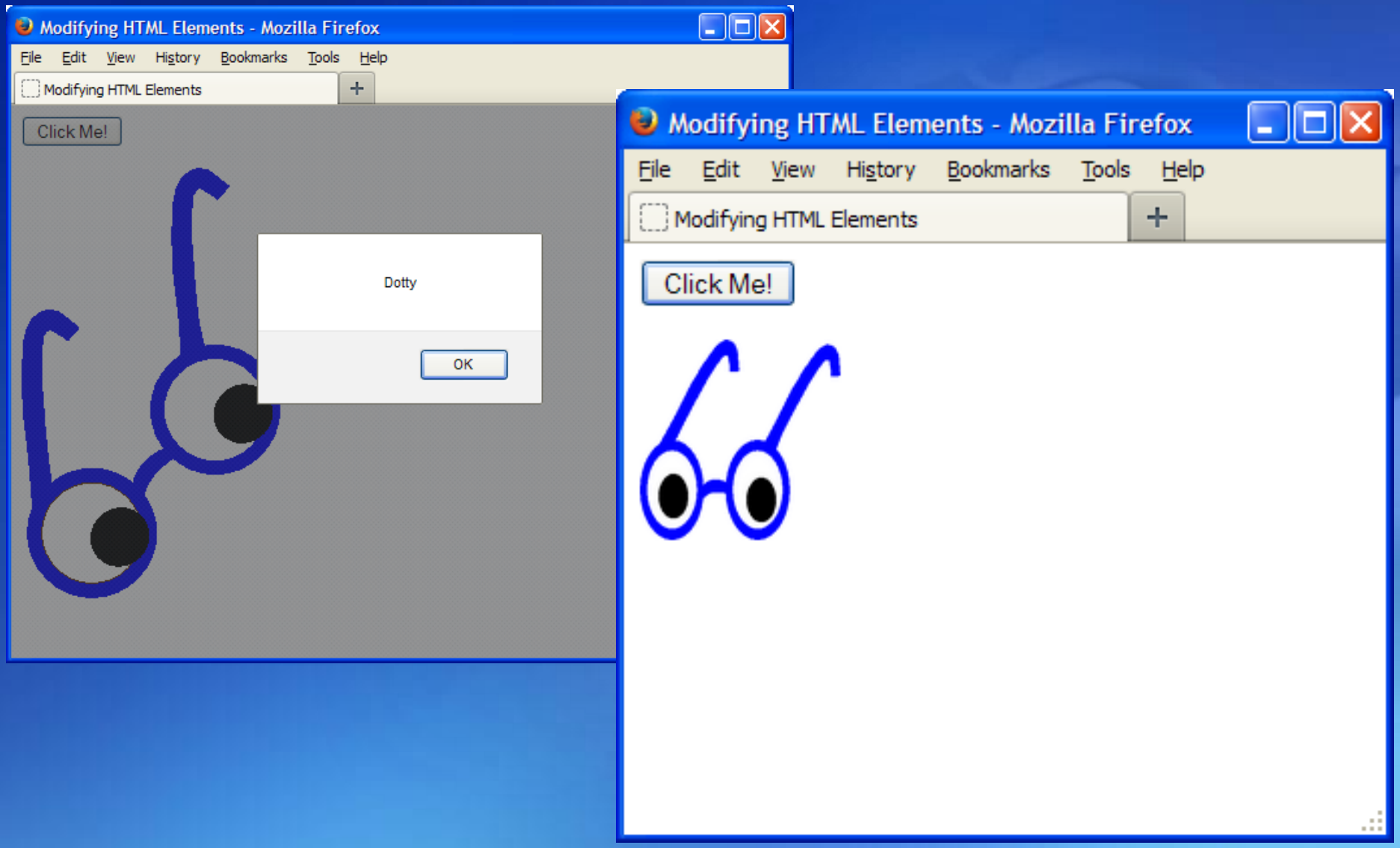
# Lab4 : Reading and modifying an image element's properties

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - imagedom.html

# Lab4 : imagedom.html

```
6       <script>
7           function changeImage() {
8               var img = document.images[0];
9
10              // get existing image attributes
11              var imgAttr = img.id + " " + img.alt + " " + img.className;
12              alert(imgAttr);
13
14              // modify
15              img.src="images/upright.gif";
16              img.width="100";
17              img.height="100";
18              img.alt="Alternative";
19              img.align="left";
20              img.title="Upright";
21          }
22      </script>
23  </head>
24  <body>
25      <button type="button" onclick="changeImage()">Click Me!</button><br>
26      <p><img src="images/dotty.gif" alt="Dotty"></p>
```

# Lab4 : Result


Modifying HTML Elements - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Modifying HTML Elements
Click Me!
Dotty
OK
Modifying HTML Elements - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Modifying HTML Elements
Click Me!

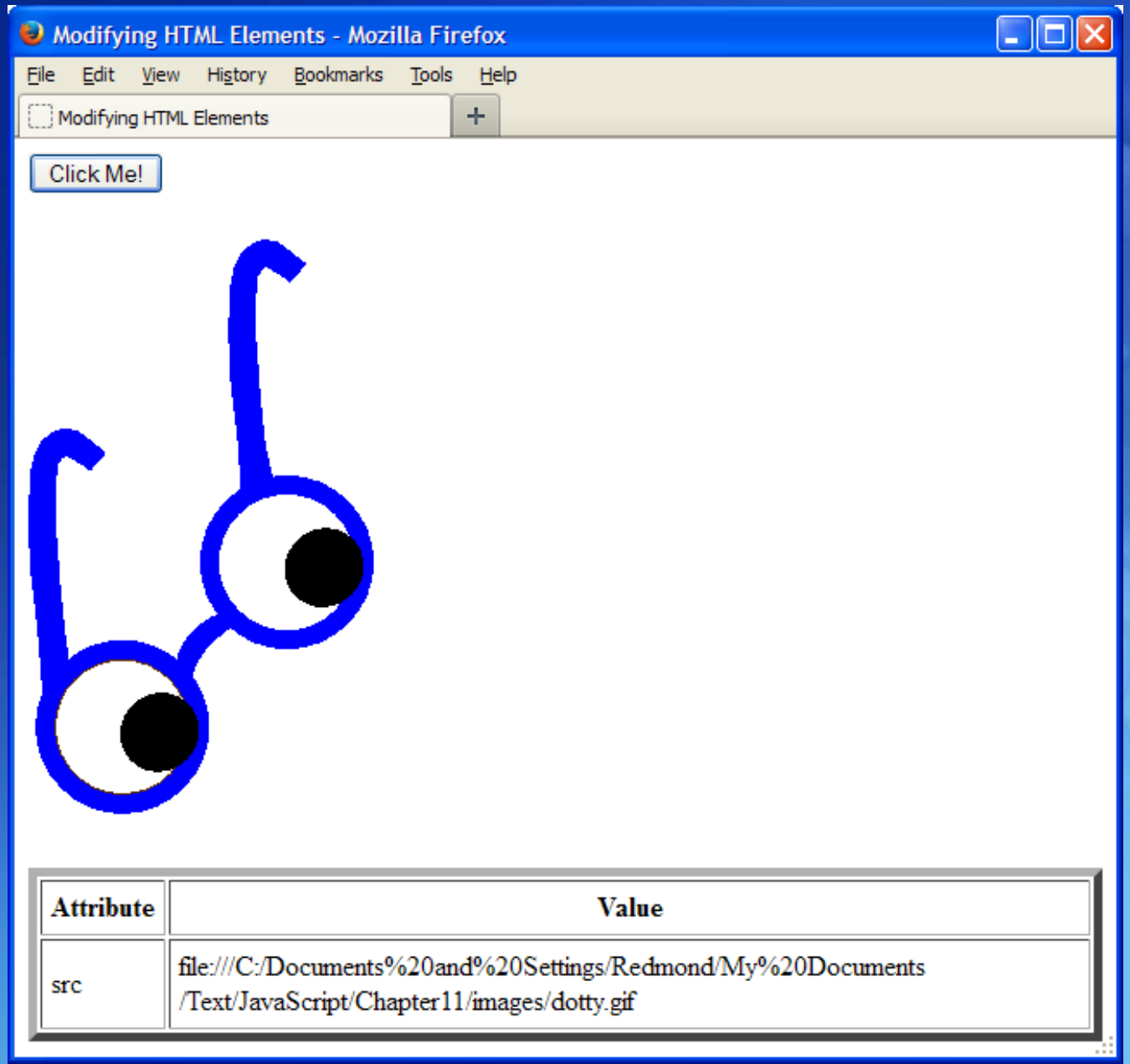
# Lab5 : Outputting image properties to a table using DOM HTML interface

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - imagedom1.html

# Lab5 : imagedom1.html

```
6       <script>
7         function changeImage() {
8           // get table and image
9           var tbl = document.getElementById('table1');
10          tbl.border="5px";
11          tbl.cellPadding="5px";
12          var img = document.getElementById("img1");
13          img.vspace="10";
14          // for each attribute, add table row
15          var row1 = tbl.insertRow(-1);
16          // create two table cells
17          var cell1 = row1.insertCell(0);
18          var cell2 = row1.insertCell(1);
19          // create text values
20          var txtAttr1 = document.createTextNode("src");
21          var txtAttr1Val = document.createTextNode(img.src);
22          // append to text values to cells
23          cell1.appendChild(txtAttr1);
24          cell2.appendChild(txtAttr1Val);
25        }
26      </script>
27    </head>
28    <body>
29      <button type="button" onclick="changeImage()">Click Me!</button><br>
30      <p><img src="images/dotty.gif" id="img1" /></p>
31      <table id="table1">
32        <tr><th>Attribute</th><th>Value</th></tr>
33      </table>
```

# Lab5 : Result


Modifying HTML Elements - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Modifying HTML Elements
Click Me!
Attribute
Value
src
file:///C:/Documents%20and%20Settings/Redmond/My%20Documents
/Text/JavaScript/Chapter11/images/dotty.gif

# DOM (HTML) Collections

- The collections of objects that you can access through the parent element, such as `document` or `form`, are represented by the `HTMLCollectioninterface`.
- The `HTMLCollectioninterface` has one property, `length`, and two methods.
  - `item` : takes a number index
  - `namedItem` : takes a string.
- Both methods return specific objects in the collection.

# DOM (HTML) Collections (Cont.)

```
var img = document.getElementById('img1');
var img = document.images.item(0);
var numImages = document.images.length;
var img =
   document.images.namedItem('original');
```

- However, a more common and simpler approach to getting an element by an identifier is to use `document.getElementById`.
- Because the use of the `id` attribute is more universal, whereas the use of `name` is heavily restricted.

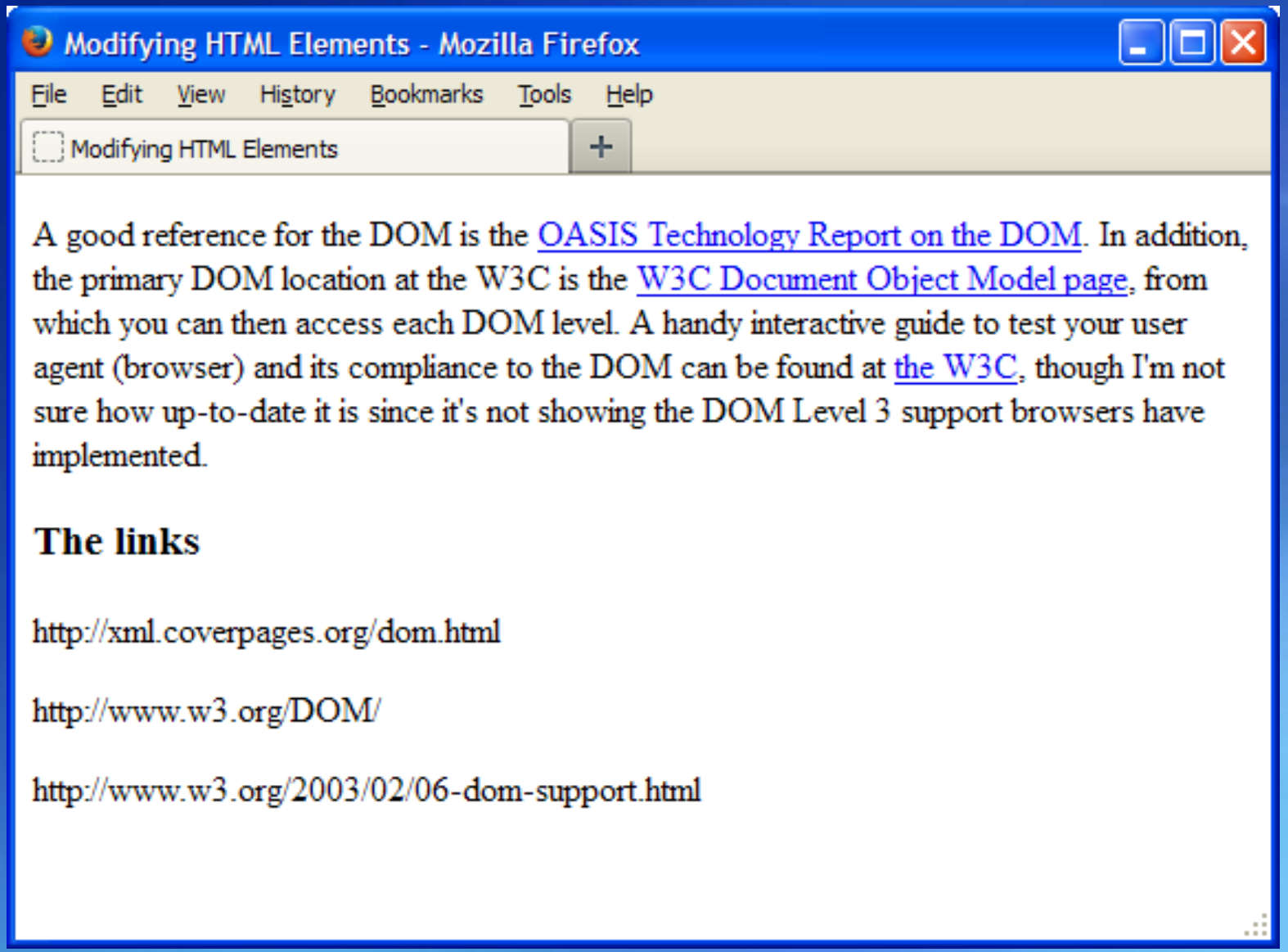
# Lab6 : Using the links collection to print out a list of URLs in a page

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - linkdom.html

# Lab6 : linkdom.html

```
6       <script>
7          window.onload=function() {
8             // get the links and the div element
9             var theLinks = document.links;
10            var theHrefs = document.getElementById("hrefs");
11            // for each link
12            for (var i = 0; i < theLinks.length; i++) {
13               // get href
14               var href = theLinks.item(i).href;
15               // create text node from href, append to new paragraph element
16               var p = document.createElement("p");
17               var txt = document.createTextNode(href);
18               p.appendChild(txt);
19               // append new paragraph to div element
20               theHrefs.appendChild(p);
21            }
22         }
23      </script>
24   </head>
25   <body>
26      <p>A good reference for the DOM is the <a
27      href="http://xml.coverpages.org/dom.html">OASIS Technology Report
28      on the DOM</a>. In addition, the primary DOM location at the W3C is the
29      <a href="http://www.w3.org/DOM/">W3C Document Object Model page</a>,
30      from which you can then access each DOM level. A handy interactive guide to
31      test your user agent (browser) and its compliance to the DOM can be found at
32      <a href="http://www.w3.org/2003/02/06-dom-support.html">the W3C</a>,
33      though I'm not sure how up-to-date it is since it's not showing
34      the DOM Level 3 support browsers have implemented.</p>
35      <div id="hrefs">
36         <h3>The links</h3>
37      </div>
```

# Lab6 : Result


Modifying HTML Elements - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Modifying HTML Elements
A good reference for the DOM is the OASIS Technology Report on the DOM. In addition, the primary DOM location at the W3C is the W3C Document Object Model page, from which you can then access each DOM level. A handy interactive guide to test your user agent (browser) and its compliance to the DOM can be found at the W3C, though I'm not sure how up-to-date it is since it's not showing the DOM Level 3 support browsers have implemented.
The links
http://xml.coverpages.org/dom.html
http://www.w3.org/DOM/
http://www.w3.org/2003/02/06-dom-support.html

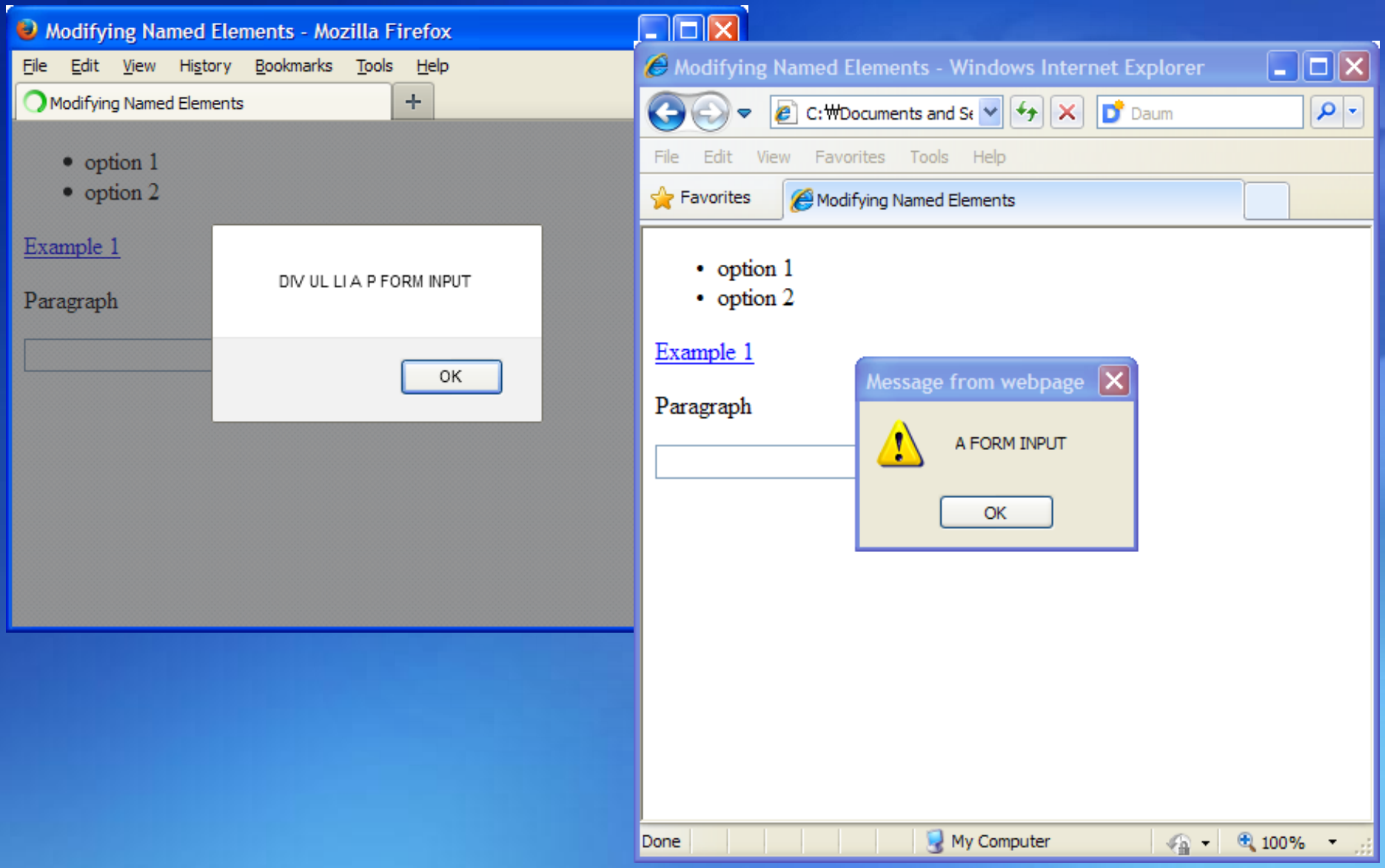
# Lab7 : Accessing HTML Objects

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - accessElements.html

# Lab7 : accessElements.html

```
6       <script>
7         window.onload = function() {
8           var typeStr = "";
9           // get all elements named 'elem' + number
10          for (var i = 1; i <= 7; i++) {
11            var nmStr = "elem" + i;
12            var nmList = document.getElementsByName(nmStr);
13            // create string of types
14            for (var j = 0; j < nmList.length; j++) {
15              typeStr += nmList[j].tagName + " ";
16            }
17          }
18          alert(typeStr);
19        }
20      </script>
21    </head>
22    <body>
23      <div name="elem1">
24        <ul name="elem2">
25          <li>option 1</li>
26          <li name="elem3">option 2</li>
27        </ul>
28      </div>
29      <a href="ch10-02.htm" name="elem4">Example 1</a>
30      <p name="elem5">Paragraph</p>
31      <form name="elem6">
32        <input type="text" name="elem7" />
33      </form>
```

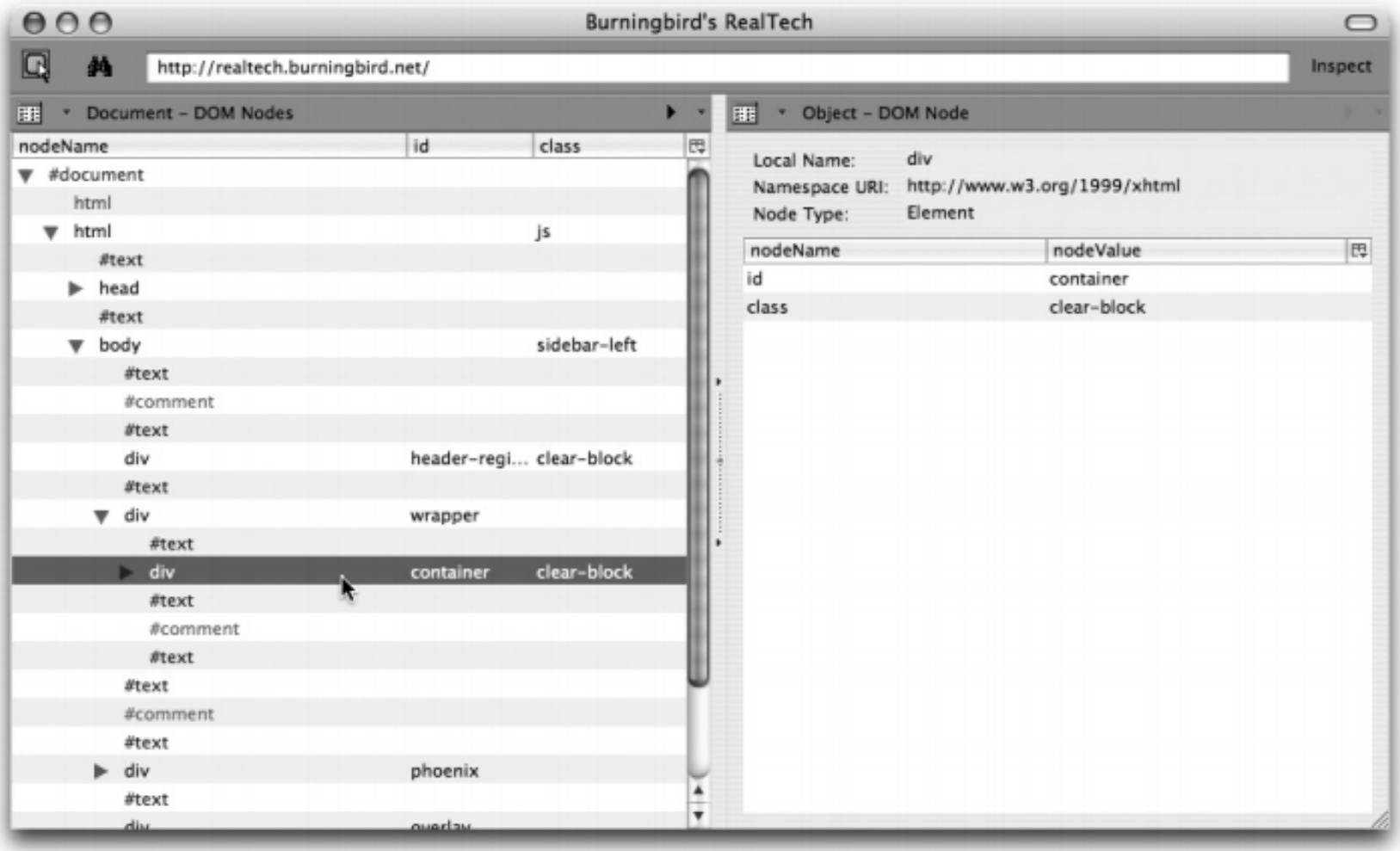
# Lab7 : Result


Modifying Named Elements - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Modifying Named Elements
option 1
option 2
Example 1
Paragraph
DIV UL LI A P FORM INPUT
OK
Modifying Named Elements - Windows Internet Explorer
C:₩Documents and Se
Daum
File Edit View Favorites Tools Help
Favorites
Modifying Named Elements
option 1
option 2
Example 1
Paragraph
Message from webpage
A FORM INPUT
OK
Done
My Computer
100%

# Understanding the DOM : The Core API

- Over the past several years, the industry has moved away from using HTML and toward the XML-based XHTML.
- The DOM HTML API is still valid for XHTML, but another set of interfaces—the DOM Core API—has gained popularity among current JavaScript developers.
- The W3C specifications for the DOM describe a document's elements as a collection of ***nodes***, connected in a hierarchical tree-like structure.

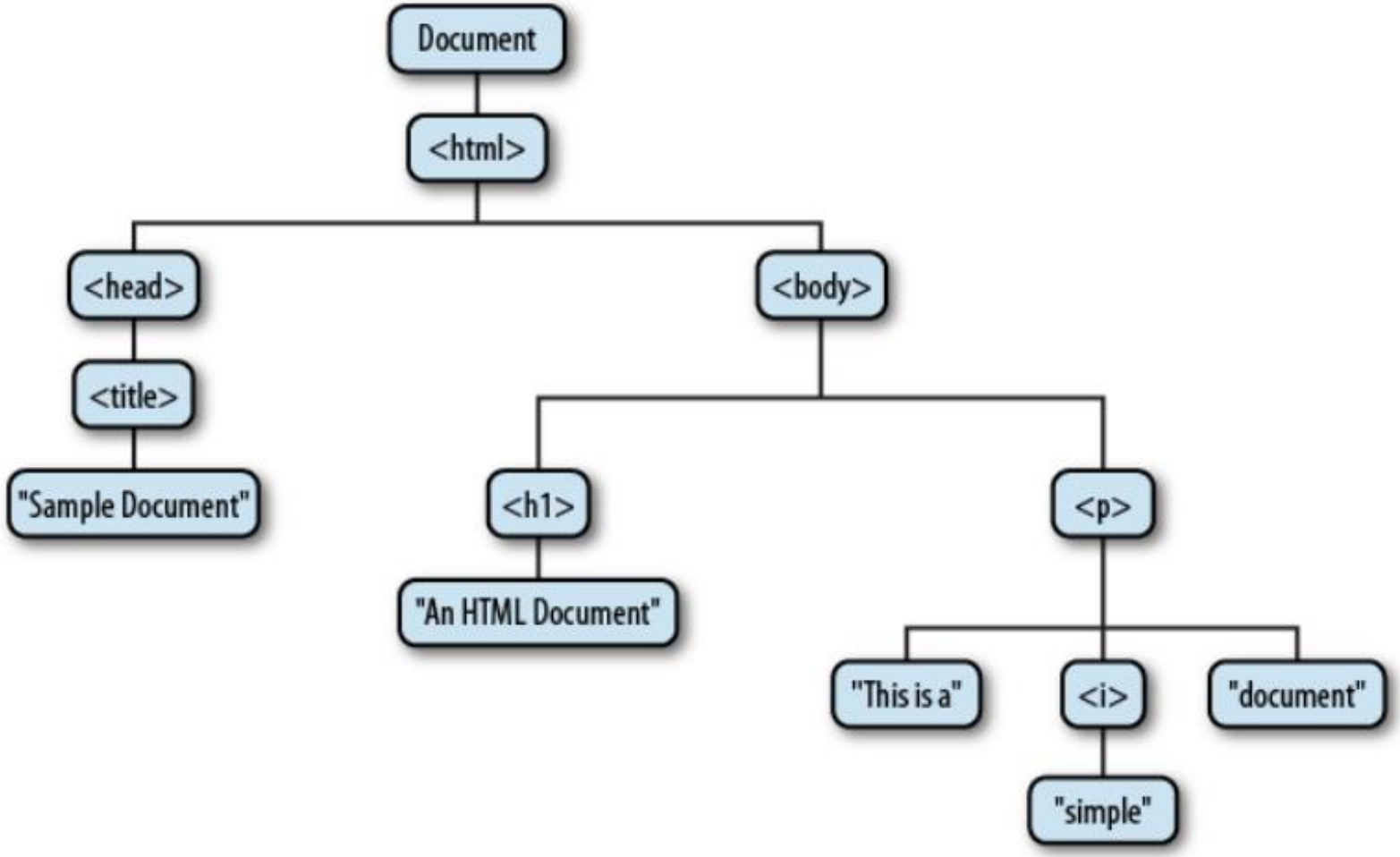
# Understanding the DOM : The Core API (Cont.)



# The DOM Tree

```
 2  <html>
 3      <head>
 4          <title> Sample Document </title>
 5      </head>
 6      <body>
 7          <h1>An HTML Document</h1>
 8          <p>This is a <i>simple</i> document.
 9      </body>
10  </html>
```

# The DOM Tree (Cont.)



# Node Properties and Methods

| Name | Description |
|---|---|
| `nodeName` | The object name, such as `HEAD` for the head element. |
| `nodeValue` | If not an element, the value of the object. |
| `nodeType` | The numeric type of the node. |
| `parentNode` | The node that is the parent to the current node. |
| `childNodes` | The `NodeList` of children nodes, if any. |
| `firstChild` | The first node in the `NodeList` children. |
| `lastChild` | The last node in the `NodeList` children. |
| `previousSibling` | If a node is a child in `NodeList`, it's the previous node in the list. |
| `nextSibling` | If a node is a child in `NodeList`, it's the next node in the list. |
| `attributes` | A `NamedNodeMap`, which is a list of key/value pairs of attributes of the element (not applicable to other objects). |
| `ownerDocument` | The owning document object, helpful in environments with more than one document. |
| `namespaceURI` | The namespace URI, if any, for the node. |
| `prefix` | The namespace prefix, if any, for the node. |
| `localName` | The local name for the node if the `namespaceURI` is present. |

# Lab8 : Accessing Node properties

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - accessNode.html

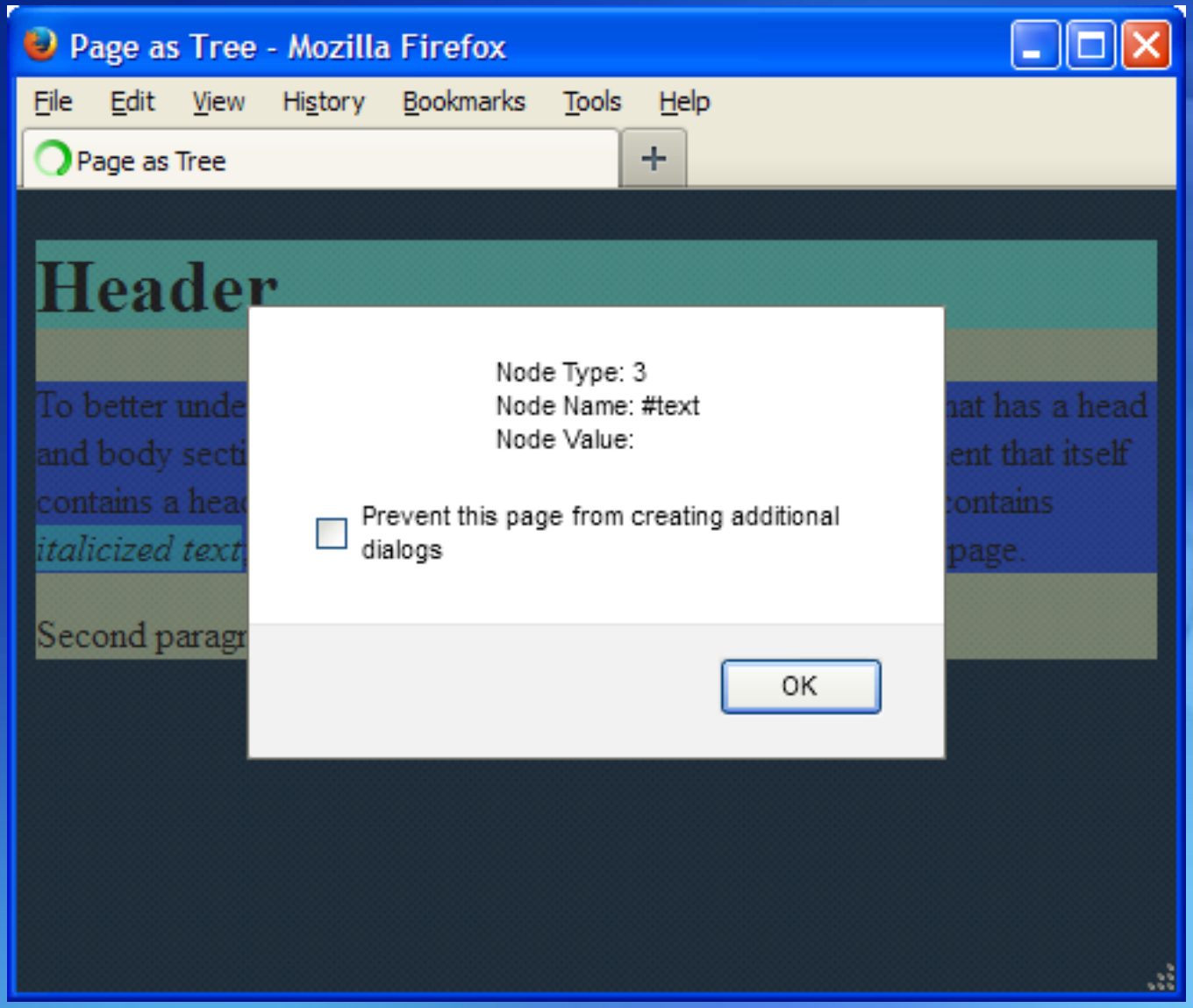
# Lab8 : accessNode.html (1/2)

```
6   <script>
7       // random color generator
8       function randomColor() {
9           var r=Math.floor(Math.random() * 255).toString(16);
10          r = (r.length < 2) ? "0" + r : r;
11          var g=Math.floor(Math.random() * 255).toString(16);
12          g = (g.length < 2) ? "0" + g : g;
13          var b=Math.floor(Math.random() * 255).toString(16);
14          b = (b.length < 2) ? "0" + b : b;
15          return "#"+r+g+b;
16      }
17      // output some node properties
18      function outputNodeProps(nd) {
19          var strNode = "Node Type: " + nd.nodeType;
20          strNode += "\nNode Name: " + nd.nodeName;
21          strNode += "\nNode Value: " + nd.nodeValue;
22          // if style set (property of Element)
23          if (nd.style) {
24              var clr = randomColor();
25              nd.style.backgroundColor=clr;
26              strNode += "\nbackgroundColor: " + clr;
27          }
28          // print out the node's properties
29          alert(strNode);
30          // now process node's children
31          var children = nd.childNodes;
32          for(var i=0; i < children.length; i++) {
33              outputNodeProps(children[i]);
34          }
35      }
```

# Lab8 : accessNode.html (2/2)

```
36          window.onload=function() {
37             outputNodeProps(document.body);
38          }
39       </script>
40    </head>
41    <body>
42       <div id="div1">
43          <h1>Header</h1>
44          <!-- paragraph one -->
45          <p>To better understand the document tree, consider a web page that
46          has a head and body section, a page title, and the body contains a div
47          element that itself contains a header and two paragraphs. One of the
48          paragraphs contains <i>italicized text</i>; the other has an image--not
49          an uncommon web page.</p>
50          <!-- paragraph two -->
51          <p>Second paragraph with image.
52             <img src="ospreythumb.jpg" alt="bird" />
53          </p>
54       </div>
55    </body>
```

# Lab8 : Result


Page as Tree - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Page as Tree
Header
Node Type: 3
Node Name: #text
Node Value:
Prevent this page from creating additional dialogs
OK

# Node Properties and Methods (Cont.)

- One property of node, `nodeType`, is a numeric.
  - `ELEMENT_NODE` : value of `1`
  - `ATTRIBUTE_NODE` : value of `2`
  - `TEXT_NODE` : value of `3`
  - `CDATA_SECTION_NODE` : value of `4`
  - `ENTITY_REFERENCE_NODE` : value of `5`
  - `ENTITY_NODE` : value of `6`
  - `PROCESSING_INSTRUCTION_NODE` : value of `7`
  - `COMMENT_NODE` : value of `8`
  - `DOCUMENT_NODE` : value of `9`
  - `DOCUMENT_TYPE_NODE` : value of `10`
  - `DOCUMENT_FRAGMENT_NODE` : value of `11`
  - `NOTATION_NODE` : value of `12`

# The DOM Core Document Object

- The `document` object is the Core interface to the web page document.
- Provides methods to create and remove page elements, as well as control.
- Provides two methods for accessing page elements: `getElementById` and `getElementsByTagName`.
- The `getElementsByTagName` method returns a list of nodes (`NodeList`) representing all page elements of a specific tag:

```
var list =
   document.getElementsByTagName("div");
```

# The DOM Core Document Object (Cont.)

- Another way to access elements is via the newly implemented `getElementsByClassName`.
- Returns a `nodeList` consisting of elements with a given class name:

```
<div class="test">...</div>
```

- To access this div element use:

```
var nds =
   document.getElementsByClassName("test");
```

# The DOM Core Document Object (Cont.)

- Another variation is to list class names separated by a space to make a list of all elements with class names equal to all entries:

```
var nds =
      document.getElementsByClassName("test test2");
```

- This is a very handy method.
- It is implemented in Safari 3.1,Opera 9.5, and Firefox 3.x and later, not IE8.

# Lab9 : Node Tree Search

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - frameset.html
  - docin.html
  - docout.html

# Lab9 : frameset.html

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Goin' for a walk</title>
5          <meta charset="utf-8">
6      </head>
7      <frameset cols="40%,*">
8          <frame name="docin" src="docin.html" />
9          <frame name="docout" src="docout.html" />
10     </frameset>
11 </html>
```

# Lab9 : docin.html

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Document In</title>
5          <meta charset="utf-8">
6      </head>
7      <body>
8          <div id="div1">
9              <h1>Header</h1>
10             <!-- paragraph one -->
11             <p>To better understand the document tree, consider a web page
12             that has a head and body section, has a page title, and contains
13             a DIV element that itself contains a H1 header and two paragraphs.
14             One of the paragraphs contains <i>italicized text</i>, the other an
15             image--not an uncommon web page.</p>
16             <!-- paragraph two -->
17             <p>Second paragraph with image.
18             <img src="images/dotty.gif" alt="dotty" /></p>
19         </div>
20 </body>
21 </html>
```

# Lab9 : docout.html

```
6   <body>
7     <script>
8       printTags(0,top.docin.document);
9
10      function printTags(domLevel,n) {
11        document.writeln("<br><br>Level " + domLevel + ":<br>");
12        document.writeln(n.nodeName + " ");
13        if (n.nodeType == 3) {
14          document.writeln(n.nodeValue);
15        }
16        if (n.hasChildNodes()) {
17          var child = n.firstChild;
18          document.writeln(" { ");
19          do {
20            document.writeln(child.nodeName + " ");
21            child = child.nextSibling;
22          } while(child);
23          document.writeln(" } ");
24          var children = n.childNodes;
25          for(var i=0; i < children.length; i++) {
26            printTags(domLevel+1,children[i]);
27          }
28        }
29      }
30    </script>
31  </body>
```

# Lab9 : Result

Goin' for a walk - Mozilla Firefox

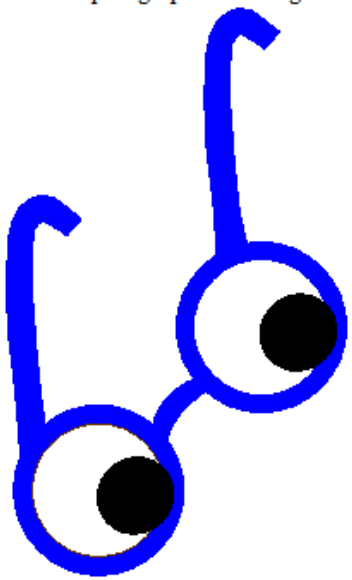
File Edit View History Bookmarks Tools Help

Goin' for a walk

## Header

To better understand the document tree, consider a web page that has a head and body section, has a page title, and contains a DIV element that itself contains a H1 header and two paragraphs. One of the paragraphs contains *italicized text*, the other an image--not an uncommon web page.

Second paragraph with image.



Level 0:
#document { html HTML }

Level 1:
html

Level 1:
HTML { HEAD #text BODY }

Level 2:
HEAD { #text TITLE #text META #text }

Level 3:
#text

Level 3:
TITLE { #text }

Level 4:
#text Document In

Level 3:
#text

Level 3:
META

Level 3:
#text

# Lab10 : A Frameset opening a sample page and an active page with script

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - frameset1.html
  - docin.html
  - docout1.html

# Lab10 : frameset1.html

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Highlighting</title>
5          <meta charset="utf-8">
6      </head>
7      <frameset cols="80%,*">
8          <frame name="docin" src="docin.html" />
9          <frame name="docout" src="docout1.html" />
10     </frameset>
11 </html>
```

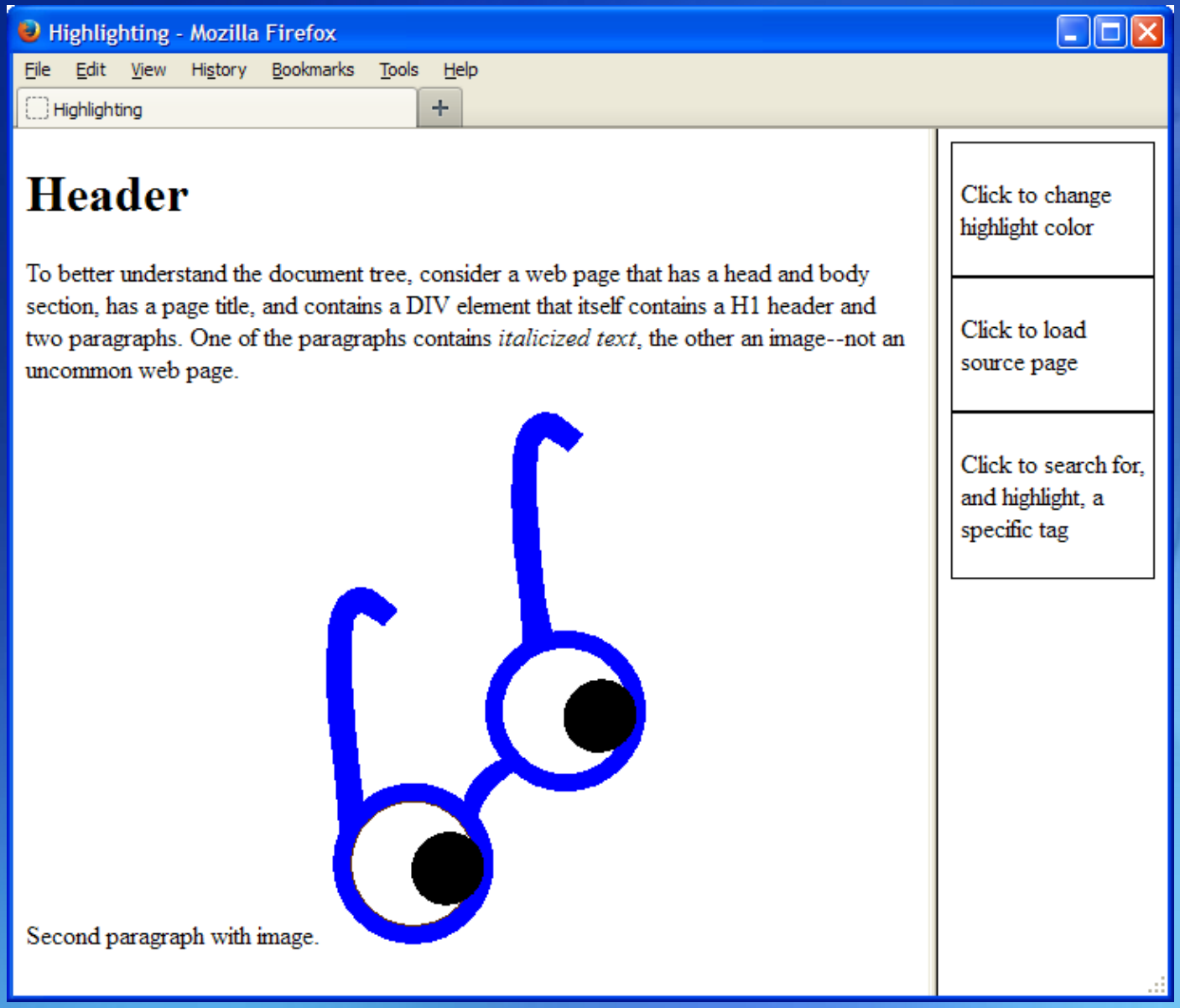
# Lab10 : docout1.html (1/2)

```
5       <meta charset="utf-8">
6       <style type="text/css">
7         div {
8           border: 1px solid #000;
9           padding: 5px;
10        }
11      </style>
12      <script>
13        var highlightColor = "#ffff00";
14        function changeColor() {
15          highlightColor=prompt("Enter highlight color (hexidecimal format)",highlightColor);
16        }
17        function loadPage() {
18          var pageURL = prompt("Enter page in this domain","");
19          top.docin.location.href=pageURL;
20        }
21        function highlightElements() {
22          var elemTag = prompt("Enter tag element name to highlight:","p");
23          var nodes = top.docin.document.getElementsByTagName(elemTag);
24          // highlight each
25          for (var i = 0; i < nodes.length; i++) {
26            var mynode = nodes[i];
27            mynode.style.backgroundColor=highlightColor;
28          }
29        }
30      </script>
31    </head>
```

# Lab10 : docout1.html (2/2)

```
32  <body>
33      <div onclick="changeColor()">
34          <p>Click to change highlight color</p>
35      </div>
36      <div onclick="loadPage()">
37          <p>Click to load source page</p>
38      </div>
39      <div onclick="highlightElements()">
40          <p>Click to search for, and highlight, a specific tag</p>
41      </div>
42  </body>
```

# Lab10 : Result



# Element and Access in Context

- Another important element in the DOM Core is a `Element`.
- All page elements within a document inherit an API and properties from `Element`.

# Element and Access in Context (Cont.)

- To do with getting and setting the attributes, or checking for the existence of attributes:
  - `getAttribute(name)`
  - `setAttribute(name,value)`
  - `removeAttribute(name)`
  - `getAttributeNode(name)`
  - `setAttributeNode(attr)`
  - `removeAttributeNode(attr)`
  - `hasAttribute(name)`

# Element and Access in Context (Cont.)

- Here's an image embedded in a web page:

```
<img src="dotty.gif" width="100"
  alt="an image" align="left" />
```

- The following code accesses the image's attributes using the `Element.getAttribute` method:

```
var img = document.images[0];
var imgStr = img.getAttribute("src") + " "
  + img.getAttribute("width") + " "
  + img.getAttribute("alt") + " "
  + img.getAttribute("align");
alert(imgStr);
```

# Element and Access in Context (Cont.)

- The following changes the value for the width and the  alt using the complementary **`Element.setAttributemethod`**:

```
img.setAttribute("width","200");
img.setAttribute("alt","This was an image");
```

# Modifying the Tree

- All three are essential objects when it comes to modifying the document tree.
- `document` :
  - Is the owner/parent of all page elements.
  - Have most factory methods for creating instances of new elements.
- `Node` :
  - Maintains the navigation within the Core API.
  - Supports the hierarchical structure of the document tree.
- `Element` :
  - Provides a way to access elements within context to apply changes to nested elements.

# Factory methods of the document object

| Method | Object created | Description |
|---|---|---|
| createElement(tagname) | Element | Creates a specific type of element |
| createDocumentFragment | DocumentFragment | The `DocumentFragment` is a lightweight document, used when extracting a section of the document tree |
| createTextNode(data) | Text | Holds any text in the page |
| createComment(data) | Comment | XML comment |
| createCDATASection(data) | CDATASection | CDATA section |
| createProcessingInstructions(target, data) | ProcessingInstruction | XML processing instruction |

# Factory methods of the document object (Cont.)

| Method | Object created | Description |
|---|---|---|
| createAttribute(name) | Attr | Element attribute |
| createEntityReference(name) | EntityReference | Creates an element to be placed later |
| createElementNS(namespace URI, qualifiedName) | Element | Creates a namespaced element |
| createAttributeNS(namespace URI, qualifiedName) | Attr | Creates a namespaced attribute |

# Modifying the Tree (Cont.)

- To create a new node, such as a text node :

```
var txtNode =
    document.createTextNode("This is a new
  text node");
```

# Modifying the Tree (Cont.)

- To add a new paragraph and text to the new div element :

```
var newDiv = document.createElement("div");
var newP = document.createElement('p');
var newText = document.createTextNode('New
                                       paragraph');
newP.appendChild(newText);
newDiv.appendChild(newP);
```

# Modifying the Tree (Cont.)

- Once you've created the new node, can add it directly into an existing document tree, using the following `Node` methods:
  - `insertBefore(newChild,refChild)`
    - Inserts a new node before the existing node, given as `refChild`
  - `replaceChild(newChild,oldChild)`
    - Replaces the existing node
  - `removeChild(oldChild)`
    - Removes the existing child
  - `appendChild(newChild)`
    - Appends a child node to the document

# Lab11 : Modifying a document

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - modify.html

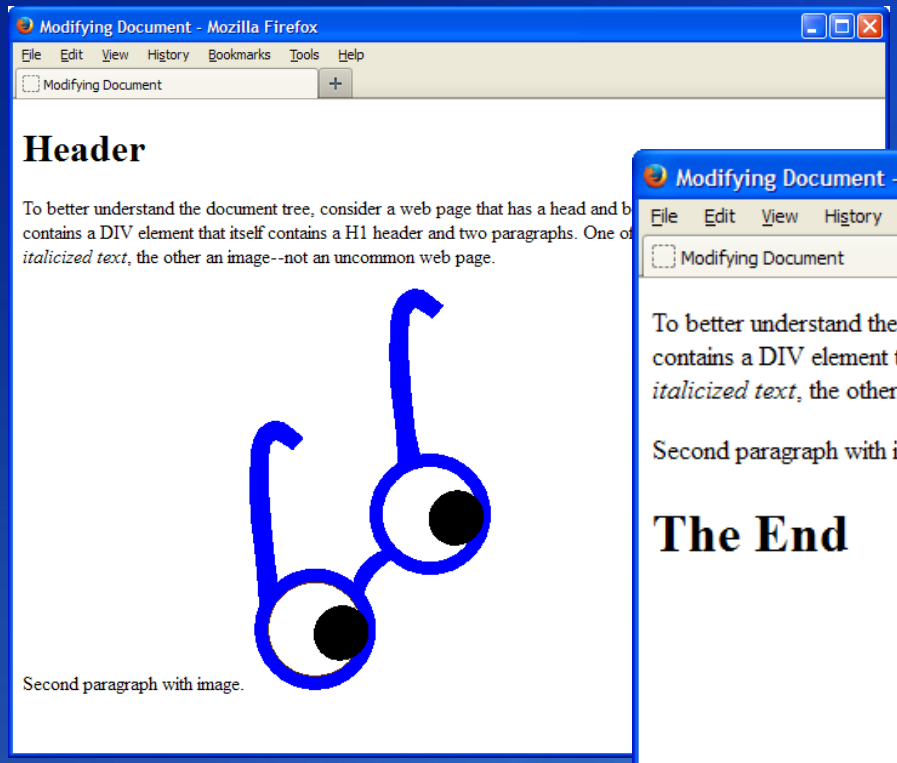
# Lab11 : modify.html (1/2)

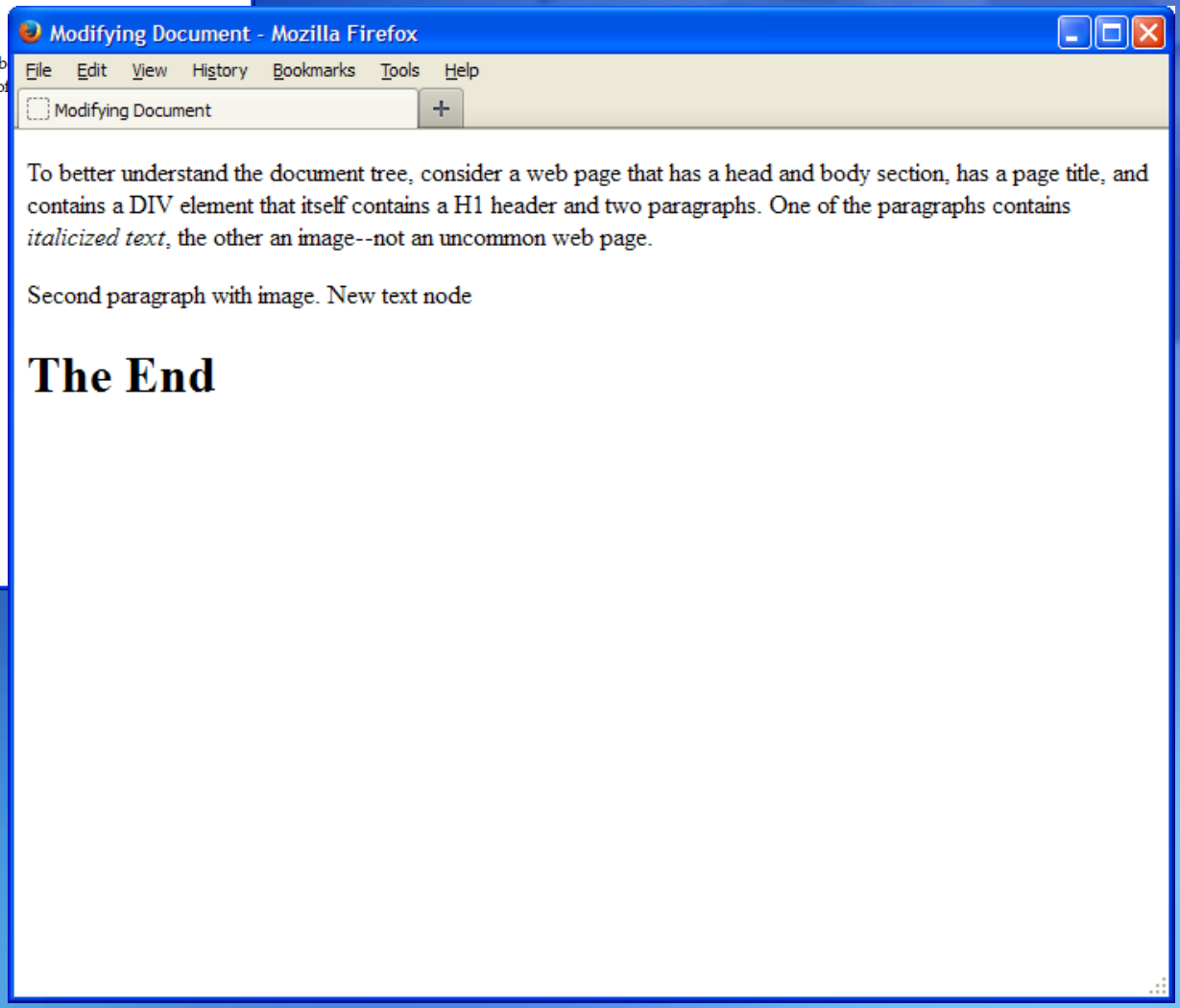
```
6   <script type="text/javascript">
7       document.onclick=changeDoc;
8       function changeDoc() {
9
10          // first, remove header
11          var hdr = document.getElementById("hdr1");
12          var div = document.getElementById("div1");
13          div.removeChild(hdr);
14
15      // replace the image with text
16          var img = document.getElementById("img1");
17          var p = document.getElementById("p2");
18          var txt = document.createTextNode("New text node");
19          p.replaceChild(txt,img);
20
21          // add new element
22          var div2= document.createElement("div");
23          div2.innerHTML="<h1>The End</h1>";
24          document.body.appendChild(div2);
25      }
26  </script>
```

# Lab11 : modify.html (2/2)

```
28    <body>
29      <div id="div1">
30        <h1 id="hdr1">Header</h1>
31        <!-- paragraph one -->
32        <p id="p1">To better understand the document tree, consider a web page
33        that has a head and body section, has a page title, and contains
34        a DIV element that itself contains a H1 header and two paragraphs.
35        One of the paragraphs contains <i>italicized text</i>, the other an
36        image--not an uncommon web page.</p>
37        <!-- paragraph two -->
38        <p id="p2">Second paragraph with image.
39        <img id="img1" src="images/dotty.gif" alt="dotty" /></p>
40      </div>
41  </body>
```

# Lab11 : Result



# Test Your Knowledge : Quiz

1. What attributes are supported for all HTML elements?

# Test Your Knowledge : Quiz

1. What attributes are supported for all HTML elements?

The attributes are `id`, `title`, `lang`, `dir`, and `className`.

# Test Your Knowledge : Quiz (Cont.)

2. Using the HTML DOM when given a named element, how would you find its element type?

# Test Your Knowledge : Quiz (Cont.)

2. Using the HTML DOM when given a named element, how would you find its element type?“two”, etc.).

Access the element’s `tagName` attribute.

# Test Your Knowledge : Quiz (Cont.)

3. Given a node in the Core DOM, how would you print out the element types of each of its children?

# Test Your Knowledge : Quiz (Cont.)

3. Given a node in the Core DOM, how would you print out the element types of each of its children?

```
var children = targetNode.childNodes;
for (var i = 0; i < children.length; i++) {
     alert(children[i].nodeType);
}
```

# Test Your Knowledge : Quiz (Cont.)

4. How would you find out the IDs (identifiers) given all `div` elements in a page?

# Test Your Knowledge : Quiz (Cont.)

4. How would you find out the IDs (identifiers) given all `div` elements in a page?

```
var divs = document.getElementsByTagName('div');
for (var i = 0; i < divs.length; i++) {
    alert(divs[i].id);
}
```

# Test Your Knowledge : Quiz (Cont.)

5. Given the following element, what are three different types of methods you could use to access this element?

```
<div id="elem1" class="thediv">...</div>
```

# Test Your Knowledge : Quiz (Cont.)

5. Given the following element, what are three different types of methods you could use to access this element?

```
<div id="elem1" class="thediv">...</div>
```

```
var theDiv = document.getElementById('elem1');

var theDivs = document.getElementsByTagName('div');
var theDiv = theDivs[0];

var theDivs = document.getElementsByClassName('thediv');
var theDiv = theDivs[0];
```

# Test Your Knowledge : Quiz (Cont.)

6. Rather than use **innerHTML**, how would you go about replacing the **header** element with a paragraph in the following **div**:

```
<div id="elem1">
    <h1>This is a header</h1>
</div>
```

# Test Your Knowledge : Quiz (Cont.)

6. Rather than use **innerHTML**, how would you go about replacing the **header** element with a paragraph in the following **div**:

```
<div id="elem1">
    <h1>This is a header</h1>
</div>
```

```
var targetElement = document.getElementById('elem1');
var specChild = targetElement.getElementsByTagName('h1')[0];
var newPara = document.createElement('p');
var paraTxt = document.createTextNode('hello');
newPara.appendChild(paraTxt);

targetElement.replaceChild(newPara,specChild);
```