Document Object Model

What are We Talking About Today?

- What is DOM?
  - It and manipulated documents
  - Represents documents as a tree structure
- DOM Levels & Evolution
  - Evolved through Levels 1-4
  - Now a Living Standard by WHATWG
- Key DOM Functions
  - getElementById, getElementsByTagName
  - Manipulate document structure, content, and styles

What is DOM

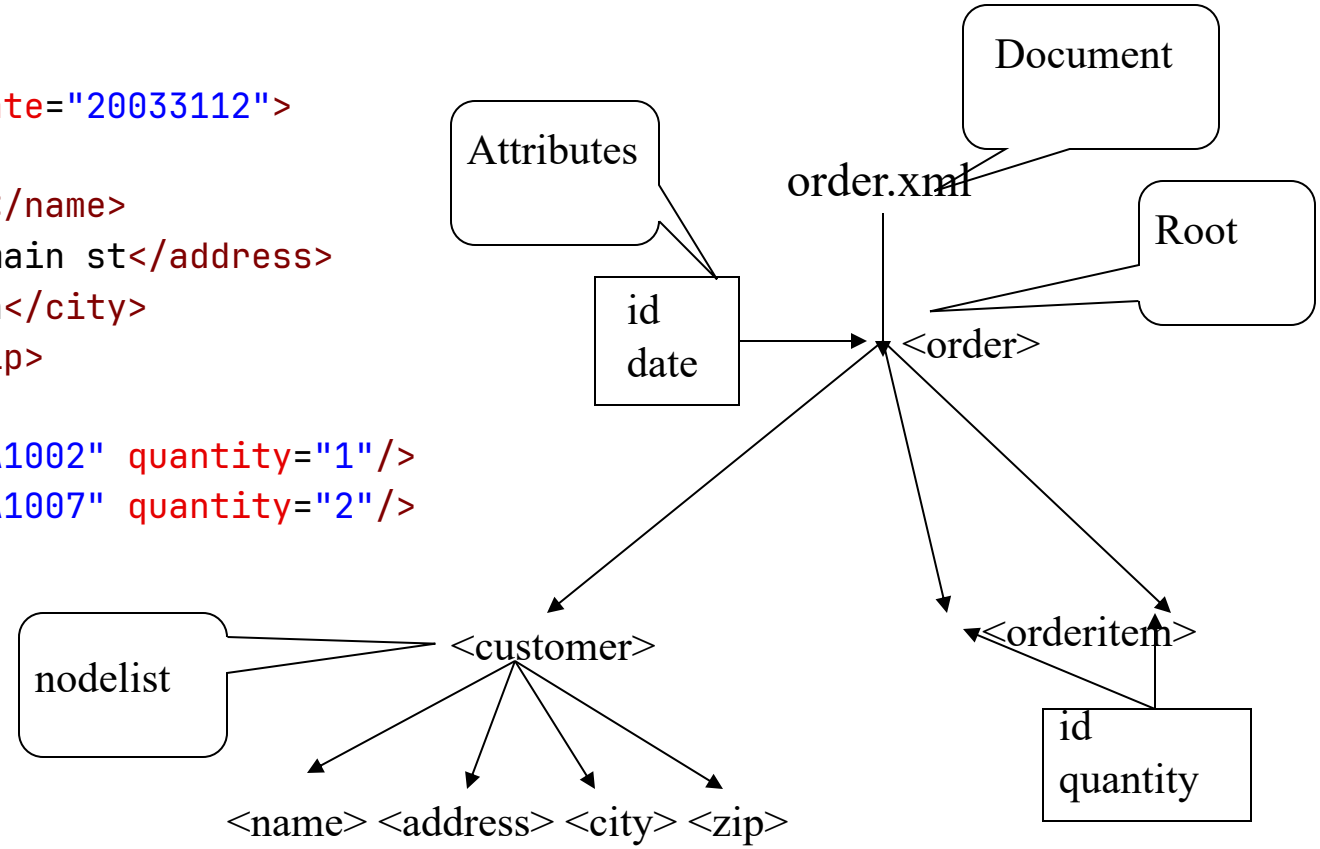
- The Document Object Model (DOM) is a programming interface for XML documents.
  - It defines the way an XML document can be accessed and manipulated
  - this includes HTML documents
- The XML DOM is designed to be used with **any programming language** and any operating system.
- The DOM represents an XML file as a tree
  - The documentElement is the top-level of the tree. This element has one or many childNodes that represent the branches of the tree.

Version History

- **DOM Level 1** concentrates on HTML and XML document models. It contains functionality for document navigation and manipulation. See:
  - http://www.w3.org/DOM/ (redirect to https://dom.spec.whatwg.org/)
- **DOM Level 2** adds a stylesheet object model to DOM Level 1, defines functionality for manipulating the style information attached to a document, and defines an event model and provides support for XML namespaces. The DOM Level 2 specification is a set of 6 released W3C Recommendations, see:
  - https://www.w3.org/DOM/DOMTR#dom2
- **DOM Level 3** consists of 3 different specifications (Recommendations)
  - DOM Level 3 Core, Load and Save, Validation, http://www.w3.org/TR/DOM-Level-3/
- **DOM Level 4 (aka DOM4)** consists of 1 specification (Recommendation)
  - W3C DOM4, http://www.w3.org/TR/domcore/
    - Consolidates previous specifications, and moves some to HTML5
- See All **DOM Technical Reports** at:
  - https://www.w3.org/DOM/DOMTR
- Now DOM specification is **DOM Living Standard (WHATWG)**, see:
  - https://dom.spec.whatwg.org

HTML or XML files viewed as a tree - order.xml

```
<order id="100" date="20033112">
  <customer>
    <name>S Spade</name>
    <address>123 main st</address>
    <city>Brooklyn</city>
    <zip>10012</zip>
  </customer>
  <orderitem id="A1002" quantity="1"/>
  <orderitem id="A1007" quantity="2"/>
</order>
```



DOM represents documents as a hierarchy of node objects
Some types of nodes have children

DOM Query Functions

- **document is the root element**

- element.querySelector(),
  element.querySelectorAll()
  - Returns subtree node/-es that match css selector
  - ..All() version returns NodeList

```
element.querySelector(".example").style.backgroundColor="red";
```

- element.getElementById("sample");
  - Returns the one location defined by id=sample, e.g. the following assigns color red to the text

```
document.getElementById("sample").style.color="rgb(255,00,00)";
```

- element.getElementsByClassName("myclass")
- element.getElementsByTagName(”font");
  - returns ALL font elements, e.g.

```
arrayOfDocFonts = document.getElementsByTagName("font");
```

Some Useful DOM manipulation methods

- **document.createElement(tagName)**
  - Only document can create nodes

```
const myEl = document.createElement("div");
```

- **.append(element) / .prepend(element)**

```
document.body.append(myEl) // adds div to the end of body
document.body.prepend(myEl) // adds div to the beginning
```

- **.replaceChildren(element1, element2, …)**

```
document.body.replaceChildren()   // clears body
document.body.replaceChildren(myEl, myEl) // replaces all body
                                          contents with two divs
```

- **.contains(el)**

```
function isInPage(node) {
  return node === document.body ? False :
document.body.contains(node);
}
```

Some Useful DOM properties

- **.innerHTML, .outerHTML, .innerText, .outerText**
  - assigns or retrieves a new value to html/text defined by id=counter2

```
document.getElementById("counter2").innerHTML = "Number of clicks = 1";
```

- **.parentNode, .childNodes, .nextSibling, …**
  - DOM tree navigation

- **.classList{.add() .remove() .replace() .toggle()}**
  - programmatic class manipulation

```
myEl.classList.add("my-class");
```

- **.style.left, .style.color, … properties**
  - one can also assign values to CSS properties, e.g.

```
document.getElementById("counter1").style.left = "500px";
```

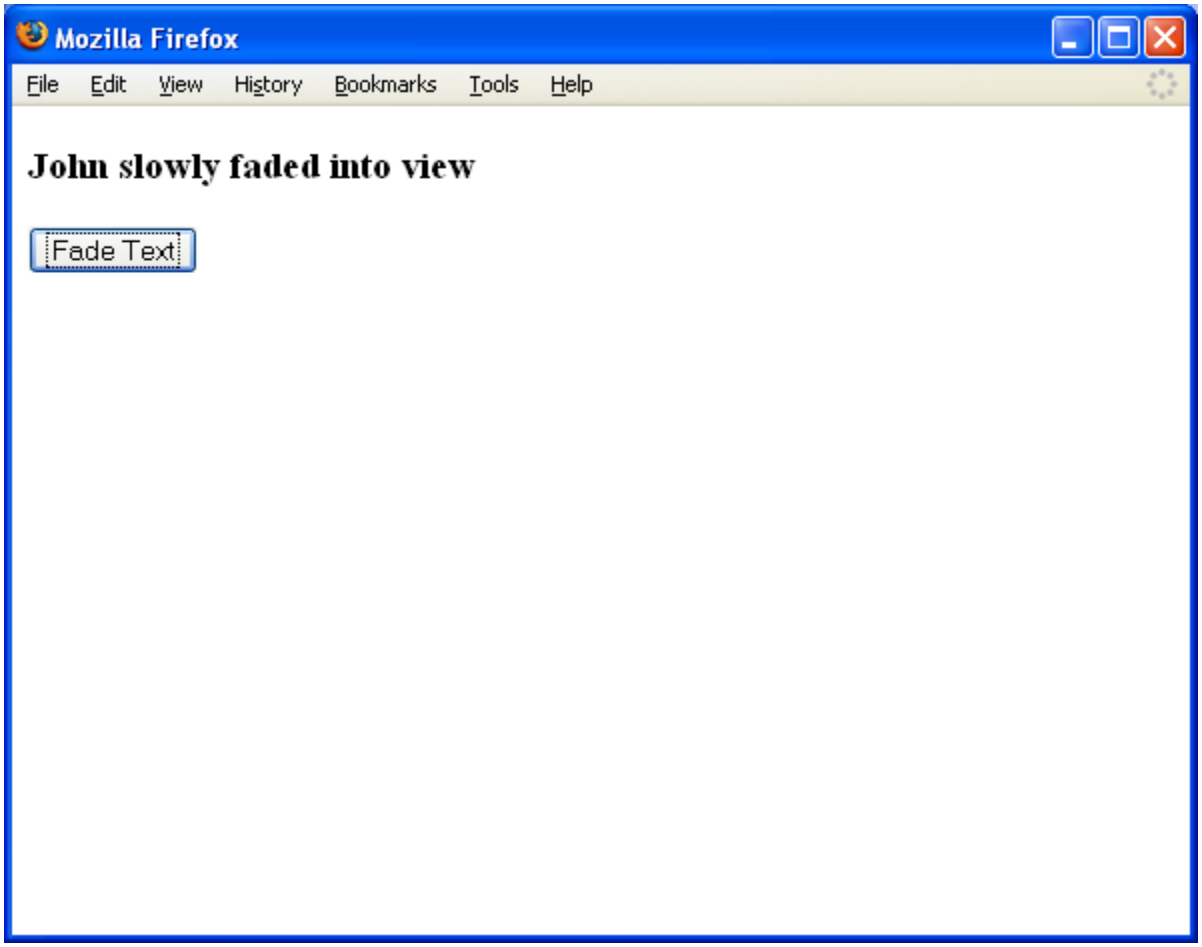
- The following slides have more examples

Example 1: Using DOM Functions to Alter a Page - Fading Text

```
<html> <head> <script>
hex = 255 // Initial color value.
function fadetext() {
  if (hex > 0) { //If color is not black yet
    hex -= 11; // increase color darkness
    const el = document.getElementById("sample");
    el.style.color = `rgb(${hex},${hex},${hex})`;
    setTimeout($ => fadetext(), 20);
  }
  else hex = 255 //reset hex value
}
</script> </head>

<body>
  <div id="sample" style="width:100%">
    <h3>John slowly faded into view</h3>
  </div>
  <button onClick="fadetext()">Fade Text</button>
</body> </html>
```

Go to: http://csci571.com/examples.html#dom

Browser Output

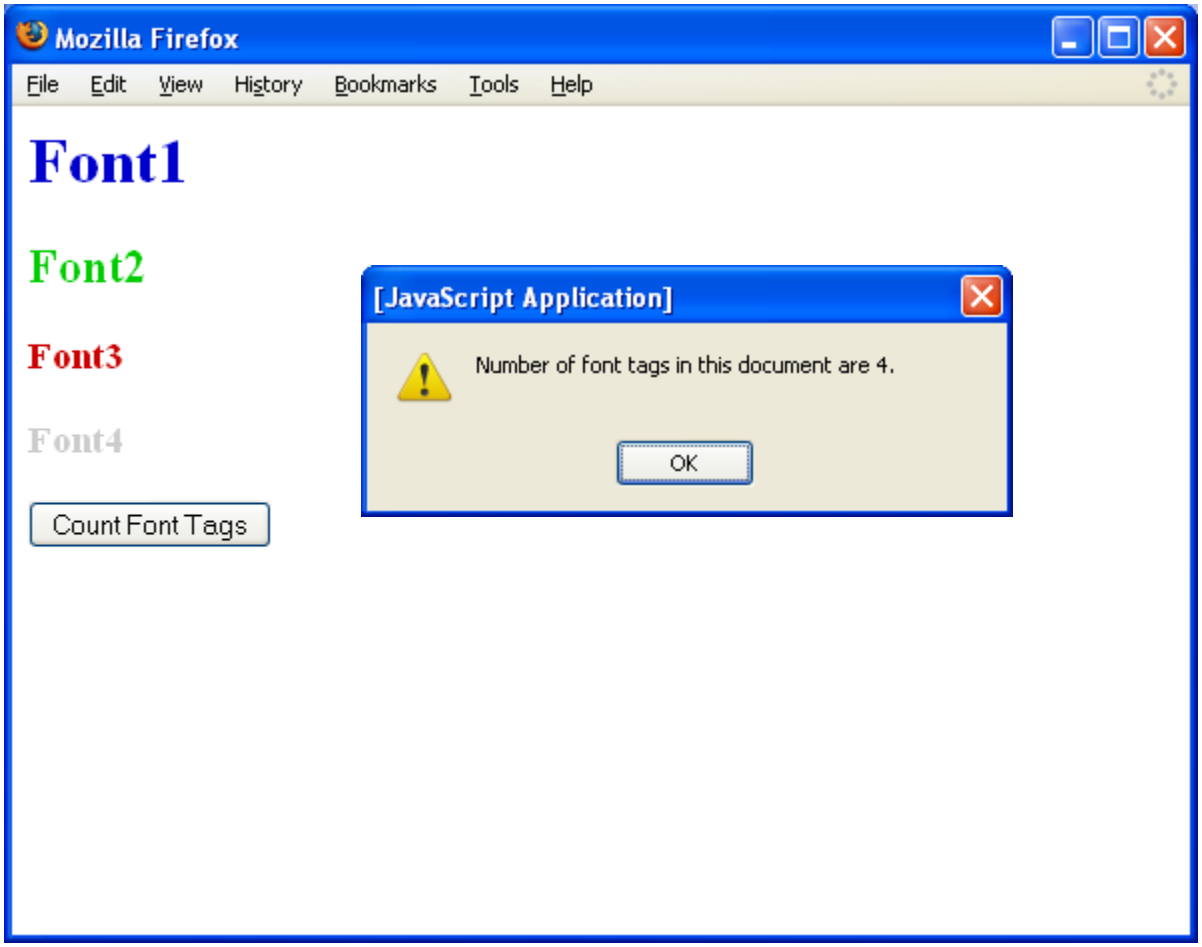


See http://csci571.com/examples/dom/ex1.html

Example 2: Extracting Elements by Tag Name

```
<html><head><script>
function handleAllTags() {
  if (!document.getElementById)
    return document.write("Unrecognized Browser Detected");

  let arrayOfDocFonts = document.getElementsByTagName("font");
                  // or document.querySelectorAll("font");
  alert(`Number of font tags here is ${arrayOfDocFonts.length}.`);
}
</script></head><body>

<h1><font color="#0000cc">Font1</font></h1>
<h2><font color="#00cc00">Font2</font></h2>
<h3><font color="#cc0000">Font3</font></h3>
<h4><font color="#cccccc">Font4</font></h4>
<input type=button onclick="handleAllTags()" value="Count Font Tags">
</body></html>
```

Browser Output



innerHTML Property

- The **innerHTML** property of an element was first introduced as non-standard extension by **Microsoft** in Internet Explorer
- Mozilla- and Gecko-based browsers (Firefox), WebKit as well as IE decided to support it, even though it was not part of the standard
- **innerHTML** is widely used in Ajax-based sites (*see later in the course*)
- Elements that do not have both an opening and closing tag cannot have an **innerHTML** property.
- The **innerHTML** property takes a string that specifies a valid combination of text and elements.
- When the **innerHTML** property is set, the given string completely replaces the existing content of the object. If the string contains HTML tags, the string is parsed and formatted as it is placed into the document
- Example 1: changes the color of the counter:

```
<DIV ID="counter2"><FONT COLOR="red">Number of clicks = 0</FONT></DIV>
```

- This line sets the innerHTML by replacing the entire text as follows:

```
document.getElementById("counter2").innerHTML = "<FONT COLOR='purple'> Number of
                                                 clicks = " + hits2 + "</FONT>";
```

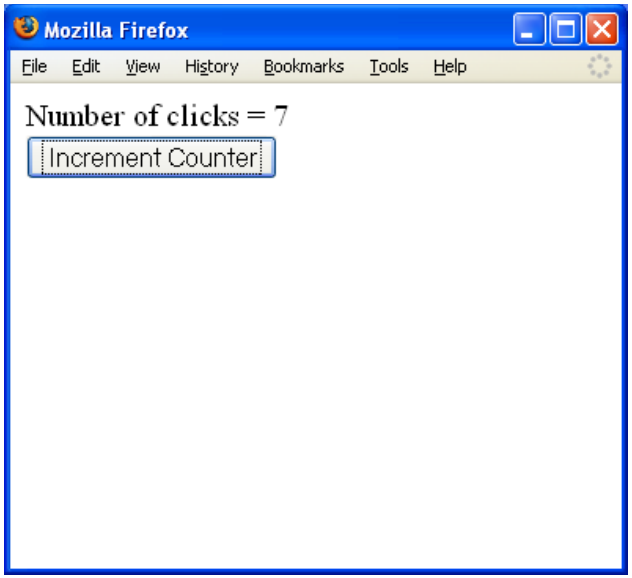
- **innerHTML has been added to the HTML5 specification (sec.8.5.4):**

  https://html.spec.whatwg.org/#the-innerhtml-property
  https://developer.mozilla.org/en-US/docs/Web/API/Element/innerHTML

Example 3: Setting innerHTML

- Example: update a counter by clicking a button

```
<div id="counter">Number of clicks = 0</div>
<input type="button" value="Increment Counter" onclick="updateMessage()">
<script>
var hits = 0;
function updateMessage() {
  hits += 1;
  const counterEl = document.getElementById("counter")
  counterEl.innerHTML = "Number of clicks = " + hits;
}
</script>
```



Final Note on innerHTML

- **Suggested Rule**: If you use innerHTML, don't use the += operator with innerHTML for the following reason:
  - Every time innerHTML is set, the HTML must be parsed, a DOM constructed and inserted into the document. This takes time.
  - For example, if elm.innerHTML has thousands of divs, tables, lists, images, etc, then calling .innerHTML += ... is going to cause the parser to re-parse *all that stuff* over again. All you want to do is append a single new element to the end.
  - innerHTML is inherently "unsafe" and the source of Cross-Site Scripting (XSS). Assign "TrustedHTML" instead of strings.
- E.g., it is better to just call **appendChild**:

```
const newElement = document.createElement("div");
newElement.innerHTML = "<p>Hello World!</p>";
elm.appendChild(newElement);
```

  This way, the existing contents of elm are not parsed again. See: https://developer.mozilla.org/en-US/docs/Web/API/Element.innerHTML
- <script> elements inserted using innerHTML do not execute (HTML5): http://www.w3.org/TR/2008/WD-html5-20080610/dom.html#innerhtml0
- appendChild() is "safer" than innerHTML for preventing XSS: JavaScript code is treated as "text" .

Example 4: Moving Objects Horizontally

- The browser-independent W3C Standard way to set and get an element's position is via the STYLE object's left and top properties
- the W3C DOM Standard defines a **"left", "right", "top", "bottom"** properties of the style object
- **E.g., Moving Objects Horizontally**

```
<input id="counter1" style="position:relative; left:0px"
  type="button" value="Move Button"
  onclick="document.getElementById('counter1').style.left = '500px'">
```

- **E.g., Moving Objects Vertically**

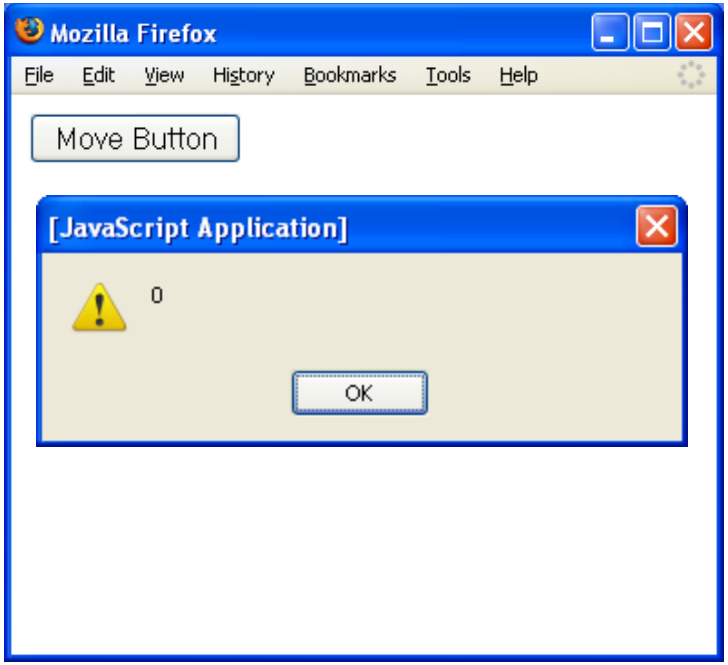
```
<input id="counter1" style="position:relative; left:0px"
  type="button" value="Move Button"
  onclick="document.getElementById('counter1').style.top = '500px'">
```

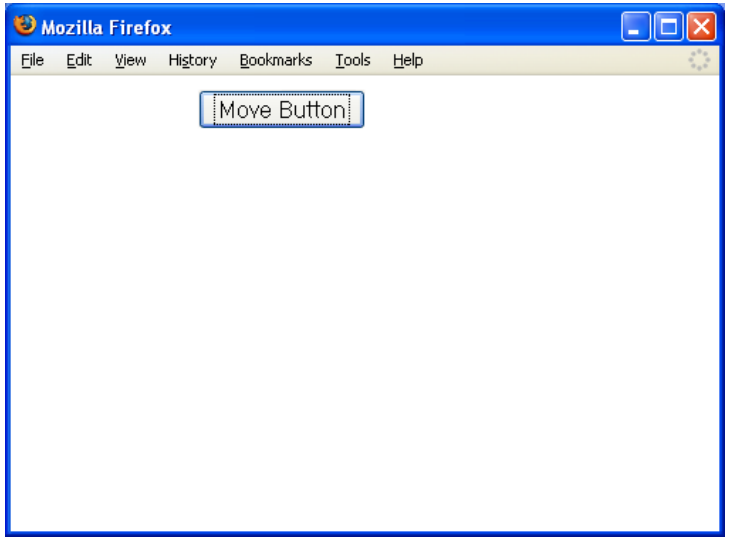
Another Example of Moving Objects on a Web Page

- The following code segment adds 50 pixels to the button's left property, every time the user clicks the button:

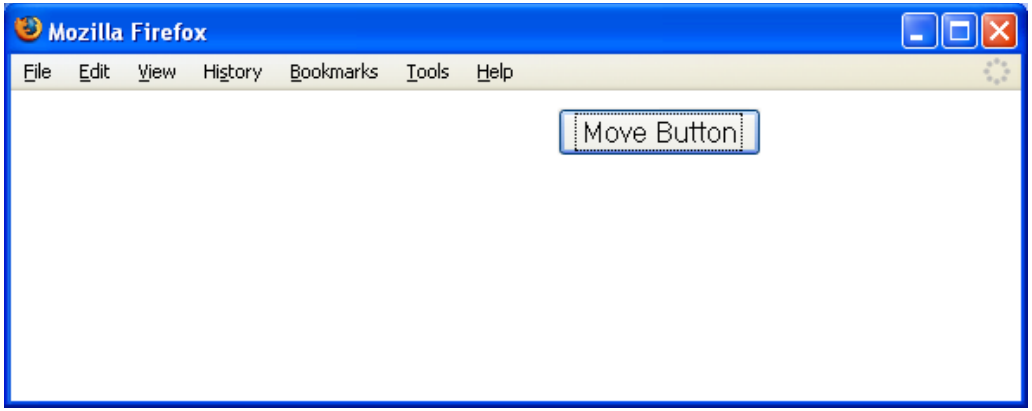
```
<input id="counter1" style="position:relative; left:0px"
   type="button" value="Move Button" onclick="handleClick()">

<script>
  var obj = document.getElementById('counter1');
  var xlocation = parseInt(obj.style.left);
  alert(xlocation);
  function handleClick() {
    xlocation += 50;
    obj.style.left = xlocation + "px";
  }
</script>
```

Browser Output





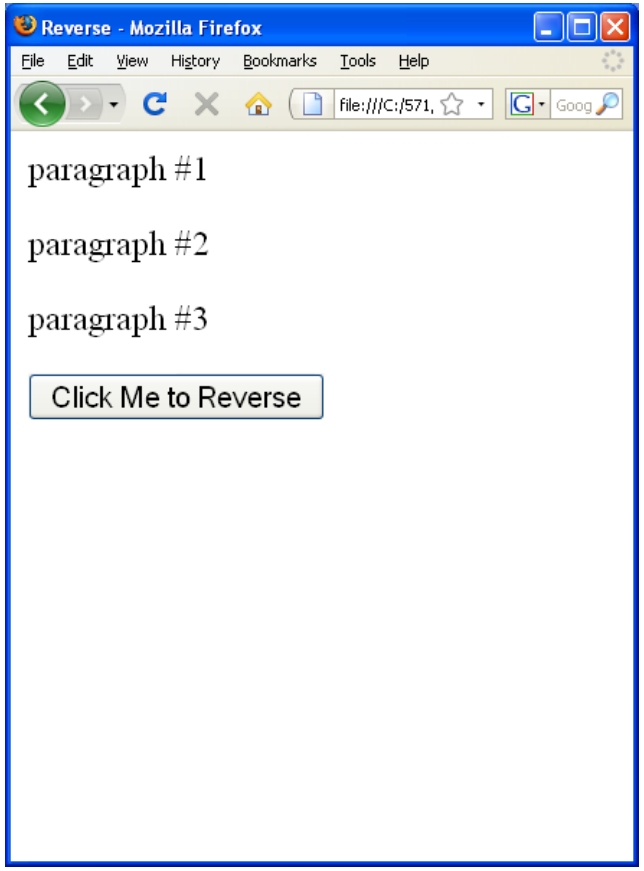
As the button is clicked
it moves to the right

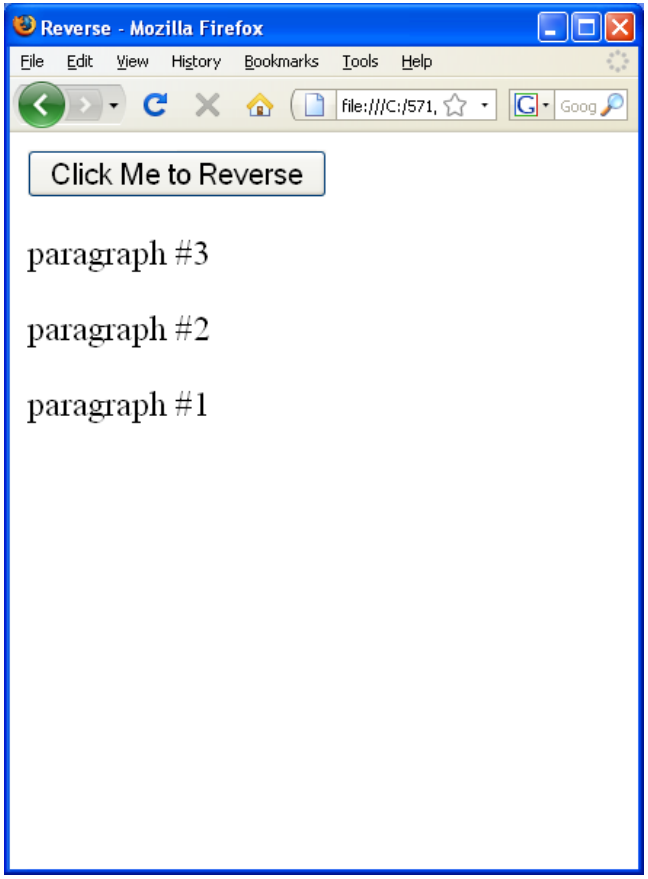


Example 5:Reversing the Nodes of a Document

```
<head><title>Reverse</title><script>
function reverse(n) { // Reverse the order of the children of Node n
  var kids = n.childNodes; // Get the list of children
  var numkids = kids.length; // Figure out how many children there are
  for (let i = numkids - 1; i >= 0; i--) // Loop backward
  {                                      through the children
    const c = n.removeChild(kids[i]); // Remove a child
    n.appendChild(c); // Put it back at its new position
  }
}
</script></head>
<body>
  <p>paragraph #1
  <p>paragraph #2
  <p>paragraph #3
  <p>
    <button onclick="reverse(document.body);">Click Me to
                                              Reverse</button>
</body>
```

Browser Output





XMLHttpRequest Object

- The XMLHttpRequest object is used to exchange data with a server
- With an XMLHttpRequest object you can:
  - Update a web page without reloading the page
  - Request data from a server after page has loaded
  - Receive data from a server after page has loaded
  - Send data to a server in the background
- All modern browsers (Edge, Firefox, Chrome, Safari, etc.) have a built-in XMLHttpRequest object.
- **"Synchronous" XMLHttpRequest is "deprecated ": being removed** from web platform (will take many years). Developer Tools will provide warning or error.
- See the open() method at:

https://xhr.spec.whatwg.org/#the-open()-method

- See **XMLHttpRequest Living Standard**:

https://xhr.spec.whatwg.org

Loading XML file into the Parser (1)

```
<script type="text/javascript">
var xmlDoc;
function loadXML(url) {
      if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
       xmlhttp=new XMLHttpRequest();
   }
 else
  {// code for IE6, IE5 [Obsolete]
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
  xmlhttp.open("GET",url,false); // 'false' = synchronous request
  xmlhttp.send();                // DEPRECATED
  xmlDoc=xmlhttp.responseXML;    // properties of XMLHttpRequest
  return xmlDoc;                 // (file returned in responseXML
 }                               // or responseText for JSON)
// ....... processing the document goes here
</script>
```

Loading XML file into the Parser

```xml
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">XQuery Kick Start</title>
    <author>James McGovern</author>
    <author>Per Bothner</author>
    <author>Kurt Cagle</author>
    <author>James Linn</author>
    <author>Vaidyanathan Nagarajan</author>
    <year>2003</year>
    <price>49.99</price>
  </book>
  <book category="web" cover="paperback">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

books.xml

Loading XML file into the Parser (2)

```
<!DOCTYPE html>
<html>
<body>
  <p id="demo"></p>
  <script>
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
      if (this.readyState == 4 && this.status == 200) myFunction(this)
    };
    xhttp.open("GET", "books.xml", true); // asynchronous request
    xhttp.send();

    function myFunction(xml) {
      var xmlDoc = xml.responseXML;
      const demoEl = document.getElementById("demo")
      demoEl.innerHTML = xmlDoc.getElementsByTagName("title")[0]
                               .childNodes[0].nodeValue;
    }
  </script>
</body>
</html>
```

This example reads "books.xml" into xmlDoc and retrieves the text value of the first <title> element in books.xml.

Node Types

Node Type	Named Constant
1	ELEMENT_NODE
2	ATTRIBUTE_NODE
3	TEXT_NODE
4	CDATA_SECTION_NODE
5	ENTITY_REFERENCE_NODE
6	ENTITY_NODE
7	PROCESSING_INSTRUCTION_NODE
8	COMMENT_NODE
9	DOCUMENT_NODE
10	DOCUMENT_TYPE_NODE
11	DOCUMENT_FRAGMENT_NODE

Another DOM Example
A simple XML file for a bookstore

```
−<bookstore>
  −<book category="cooking">
      <title lang="en">Everyday Italian</title>
      <author>Giada De Laurentiis</author>
      <year>2005</year>
      <price>30.00</price>
    </book>
  +<book category="children"></book>
  +<book category="web"></book>
  +<book category="web" cover="paperback"></book>
</bookstore>
```

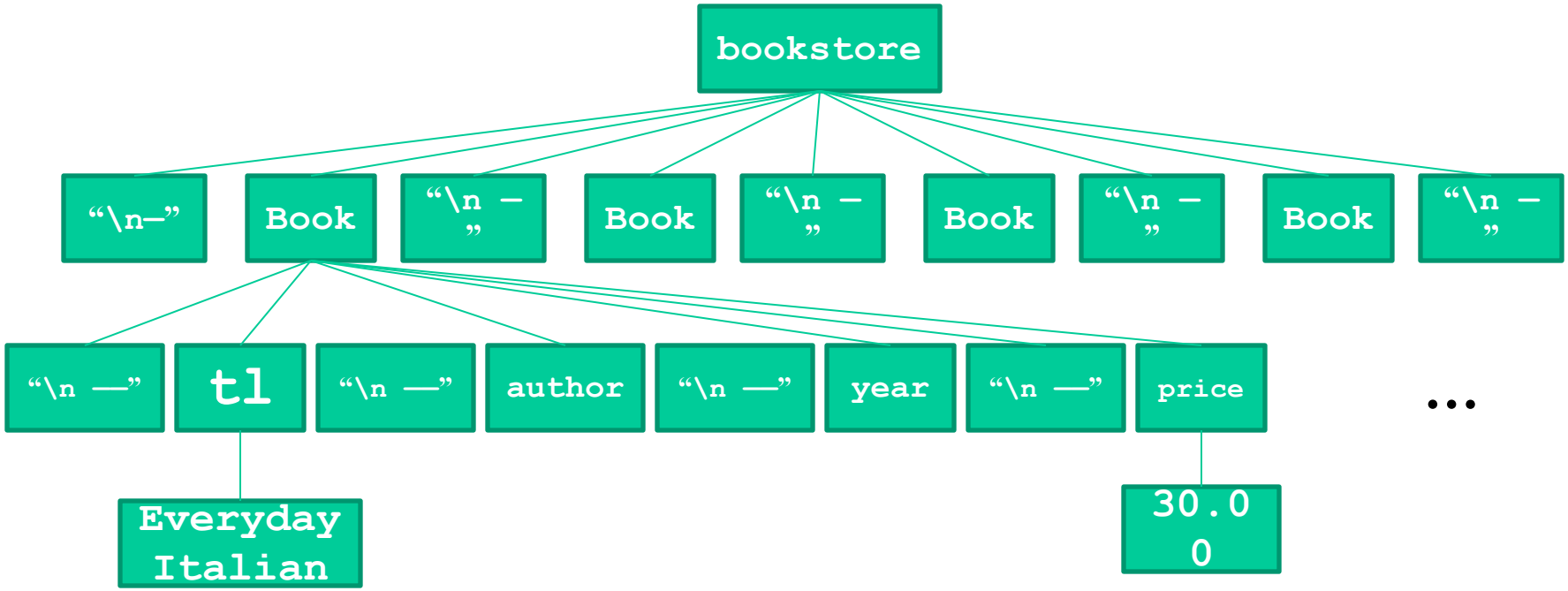

Some Node Types in an XML File

A Sample XML File

```
- <bookstore>
   <book category="cooking">
     <title lang="en">Everyday Italian</title>
     <author>Giada De Laurentiis</author>
     <year>2005</year>
     <price>30.00</price>
   </book>
 + <book>
 + <book>
</bookstore>
```

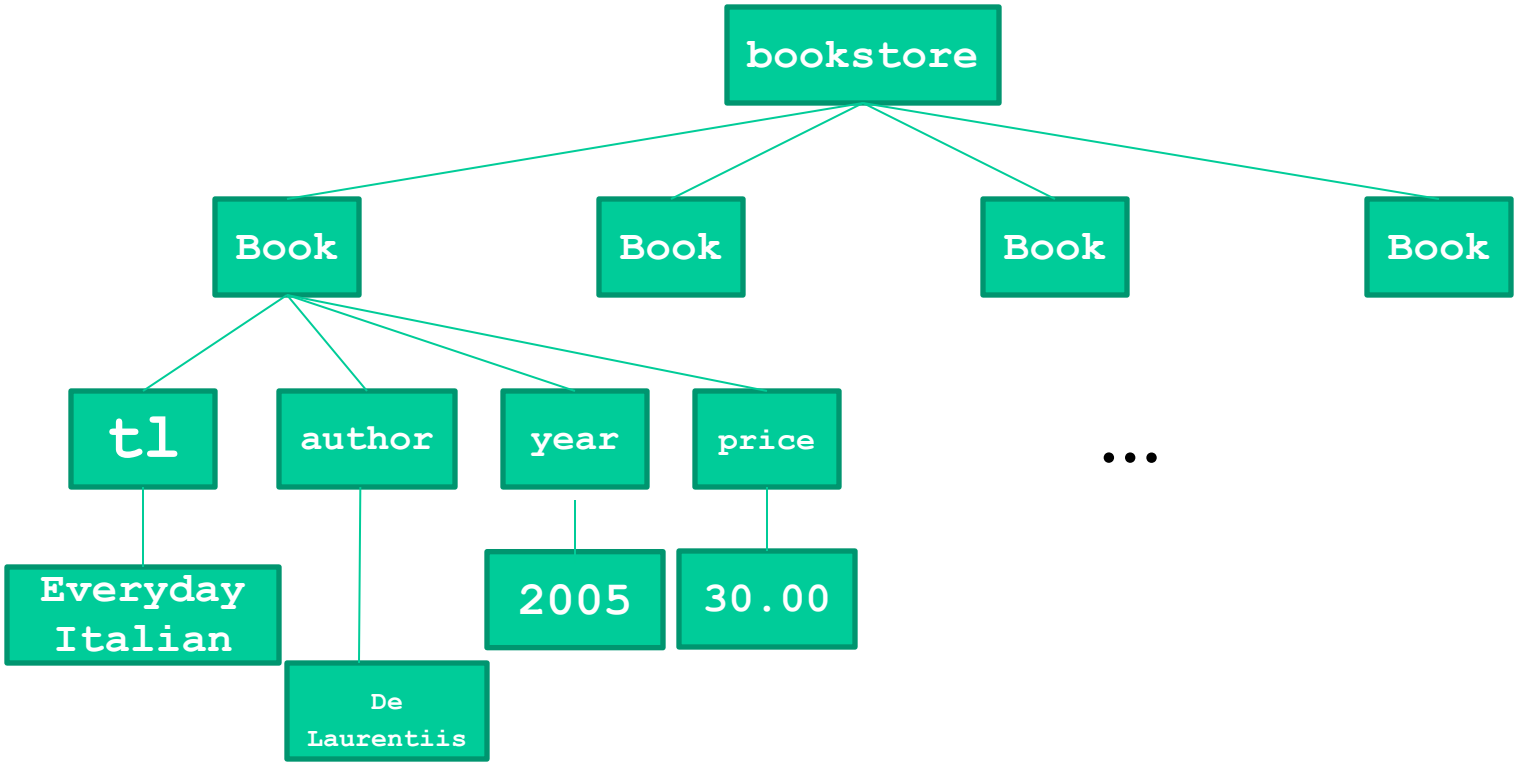
- Some possible node types
- ELEMENT_NODE (type 1)
  - bookstore, book, title, author, year, price
- TEXT_NODE (type 3)
  - "/n" nodes
  - "Everyday Italian", "30.00", …
- Hint:
  - element nodes have children
  - text nodes are leaves
- x[i].nodeType == 1
  - tests for element nodes.
  - text nodes (like "\n") are ignored

The Same XML tree in FireFox



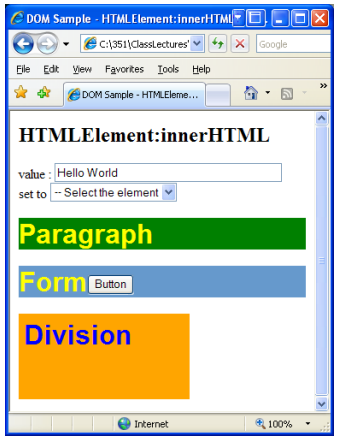
Where – represents one space character

An XML Tree in other Browsers

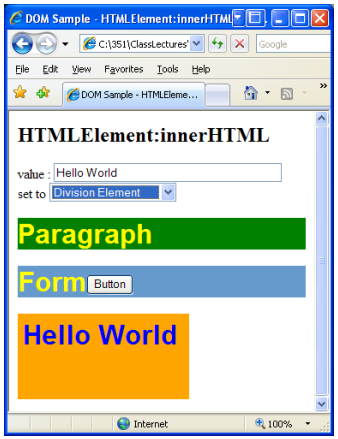


http://csci571.com/examples/dom/domtest.html

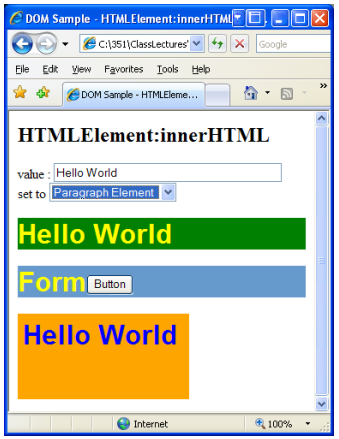
Example 6: DOM and Three InnerHTML Examples



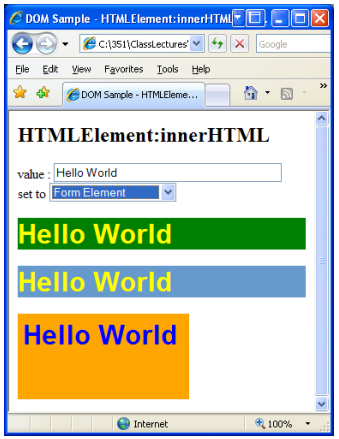
Initial page



Select division



Select paragraph



Select form

id definitions
t1 (orange)
t2 (green)
t3 (yellow)

setInnerHTML function defined here

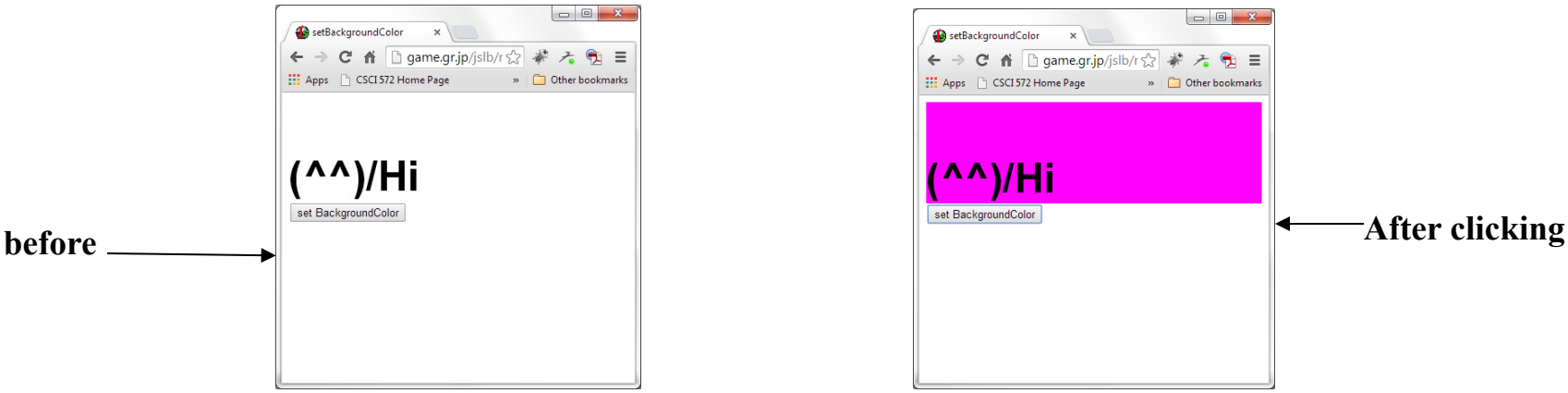
Handle selection

DOM Sample - HTMLElementinnerHTML.htm - Notepad

```
<HTML><HEAD><TITLE>DOM Sample - HTMLElement:innerHTML</TITLE>
<STYLE type=text/css>.smp {
        PADDING-RIGHT: 0.2em; PADDING-LEFT: 0.2em; PADDING-BOTTOM: 0.2em; WIDTH: 200px;
        PADDING-TOP: 0.2em; POSITION: absolute; HEIGHT: 100px}
#t1 {FONT-WEIGHT: 700; FONT-SIZE: 2em; COLOR: blue;
     FONT-FAMILY: sans-serif; BACKGROUND-COLOR: orange}
#t2 {FONT-WEIGHT: 700; FONT-SIZE: 2em; LEFT: 120px;
     COLOR: yellow; FONT-FAMILY: sans-serif; TOP: 200px; BACKGROUND-COLOR: green}
#t3 {FONT-WEIGHT: 700; FONT-SIZE: 2em; COLOR: yellow;
FONT-FAMILY: sans-serif; BACKGROUND-COLOR: #6699cc}</STYLE>
<SCRIPT language=JavaScript type=text/javascript><!--
function notSupported(){ alert('your browser is not supported.'); }
function setInnerHTML(nm,value){
  if(nm == '') return;
  var element=document.getElementById?document.getElementById(nm):(document.all?document.all(nm):null)
  if(element){
    if(element.innerHTML){
      element.innerHTML=value;
    }
    else notSupported();
  }
  else notSupported();
}
// --></SCRIPT></HEAD><BODY>
<H2>HTMLElement:innerHTML</H2>
<FORM>value : <INPUT size=40 value="Hello World" name=t><BR>set to <SELECT id=sel
onchange=setInnerHTML(this.options[this.selectedIndex].value,form.t.value);
name=sel> <OPTION value="" selected>-- Select the element<OPTION
  value=t1>Division Element<OPTION value=t2>Paragraph Element<OPTION
  value=t3>Form Element</OPTION></SELECT> </FORM>
<P id=t2>Paragraph</P>
<FORM id=t3 name=t3>Form<INPUT type=button value=Button></FORM>
<DIV class=smp id=t1>Division</DIV>
<TABLE height=100 width=250>
  <TBODY><TR><TD></TD></TR></TBODY></TABLE></BODY></HTML>
```

3 select options: division, paragraph form

Example 7: Changing Background color

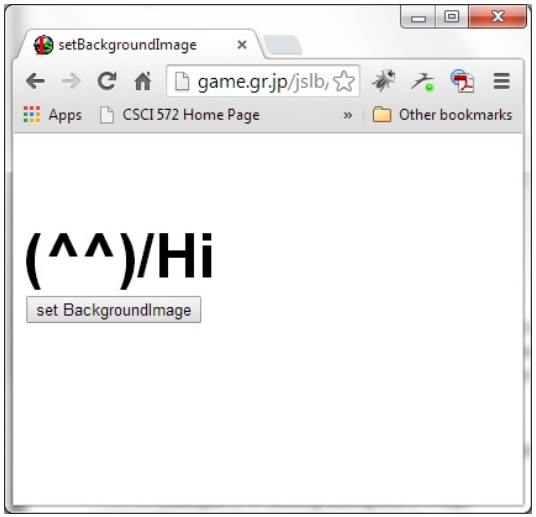


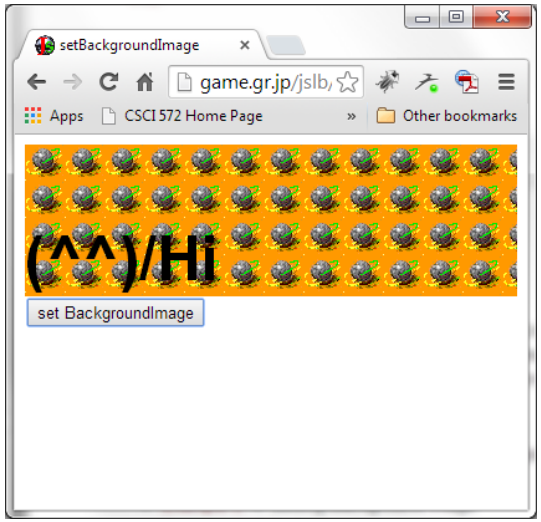

```
<HTML><HEAD><TITLE>setBackgroundColor</TITLE>
<SCRIPT>
function setBackgroundColor(id, bgcolor) {
  document.getElementById(id).style.backgroundColor = bgcolor;
}
</SCRIPT></HEAD>

<BODY>
  <DIV ID="test" STYLE="font:900 50px Arial"><BR>(^^)/Hi</DIV>
  <FORM>
    <INPUT TYPE="button" VALUE="set BackgroundColor"
           onClick="setBackgroundColor('test','magenta')">
  </FORM>
</BODY></HTML>
```

See all examples at: http://csci571.com/examples.html#dom

Example 8: Changing Background Image

before



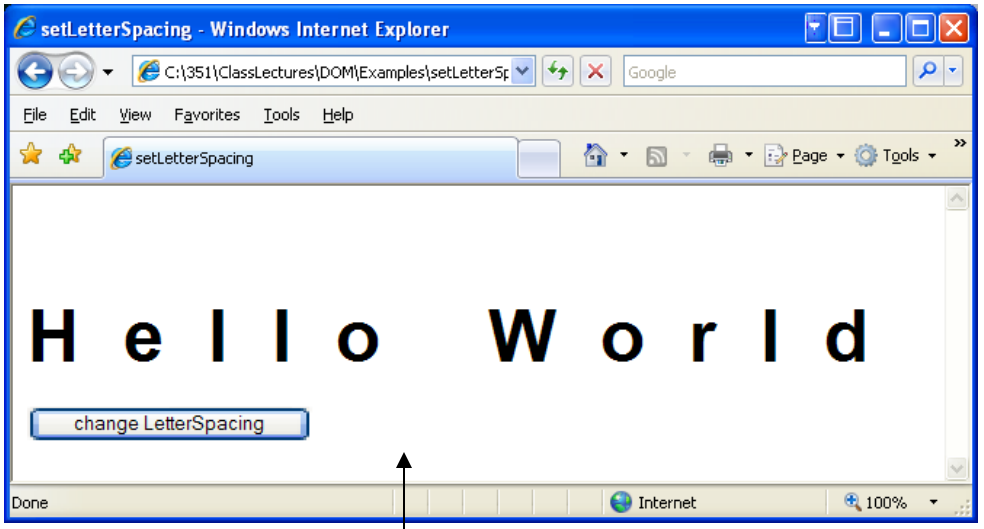


After clicking

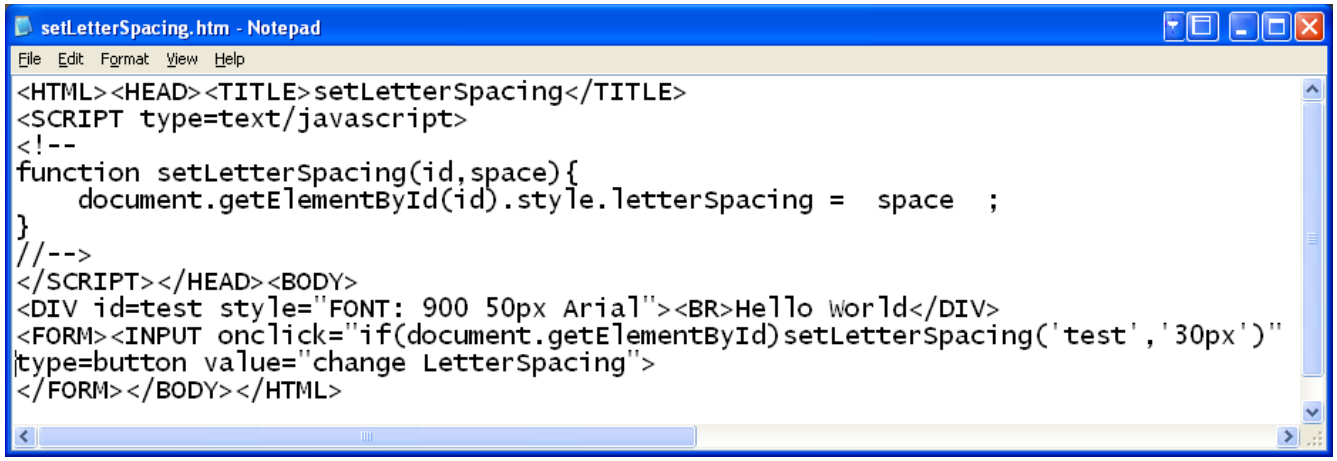
```
<HTML><HEAD><TITLE>setBackgroundImage</TITLE>
<SCRIPT>
  function setBackgroundImage(id, image) {
    document.getElementById(id).style.backgroundImage = 'url(' + image + ')';
  }
</SCRIPT></HEAD>

<BODY>
  <DIV ID="test" STYLE="font:900 50px Arial"><BR>(^^)/Hi</DIV>
  <FORM>
    <INPUT TYPE="button" VALUE="set BackgroundImage"
           onClick="setBackgroundImage('test','tamas.gif')">
  </FORM>
</BODY>
</HTML>
```

Example 10: Changing Letter Spacing

before

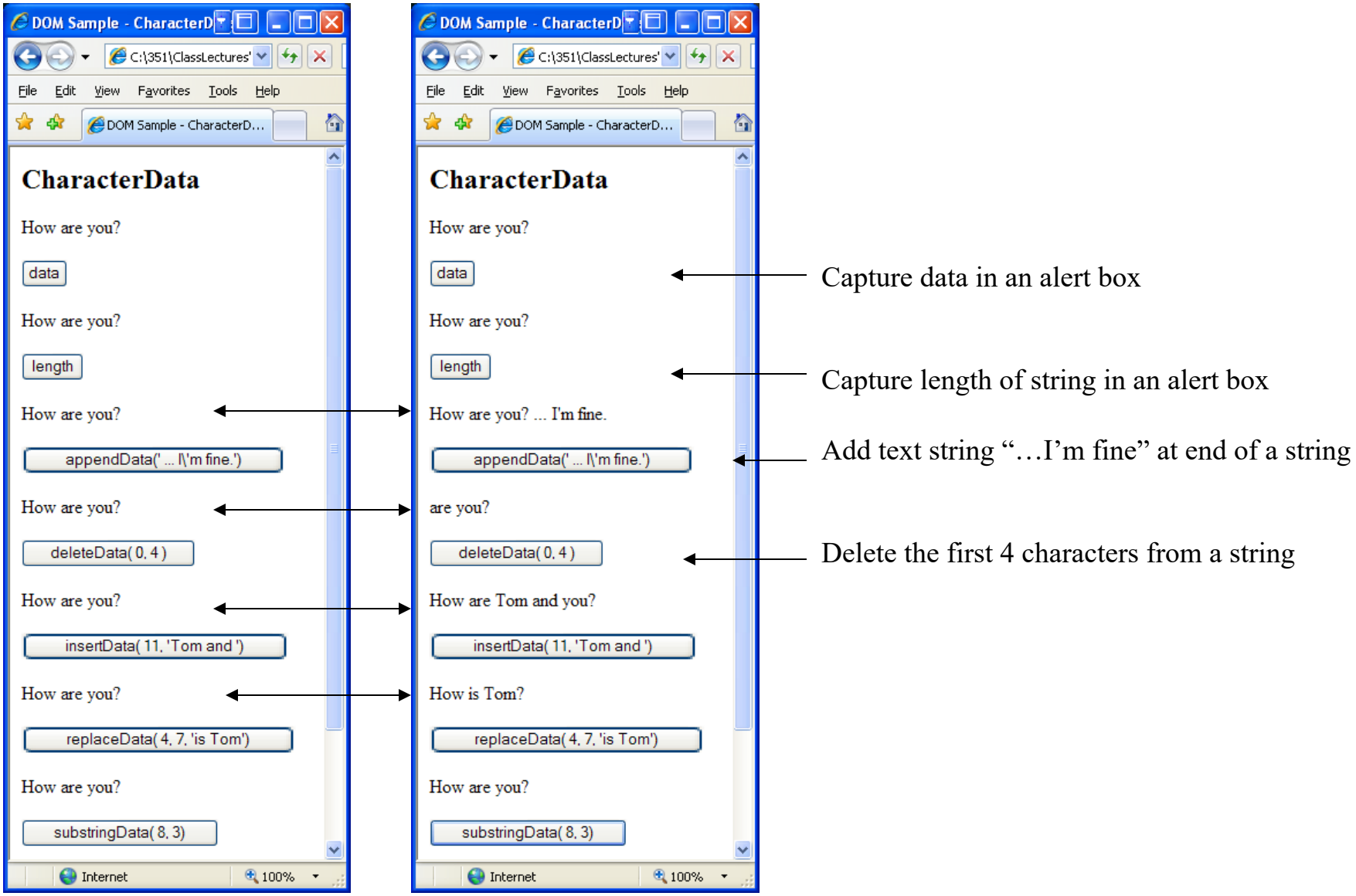


After clicking

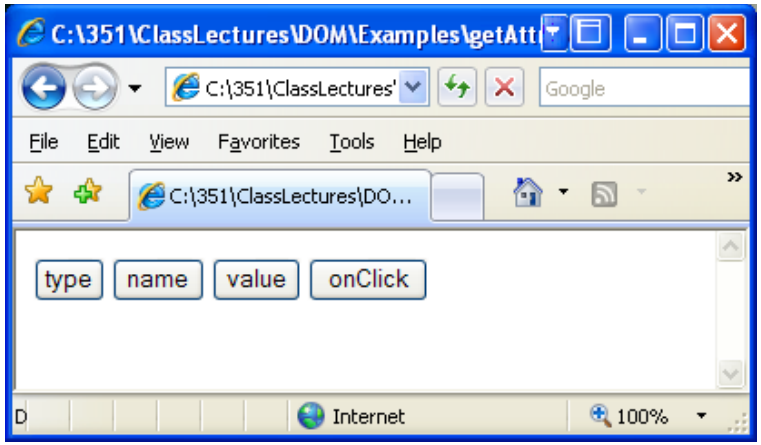


http://csci571.com/examples/dom/setLetterSpacing.htm

Example 11: DOM and Manipulating Character Data



Example 12: DOM and Retrieving Attributes



type

name

value

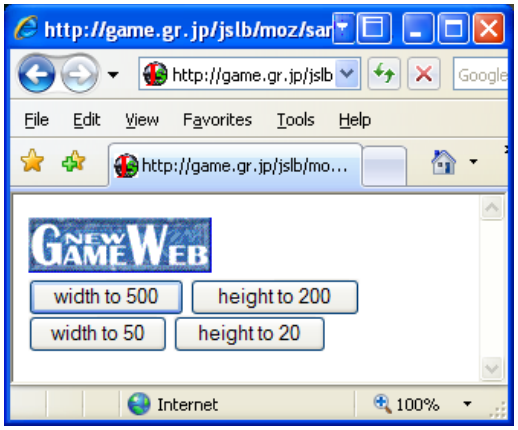
onClick

getAttribute.htm - Notepad

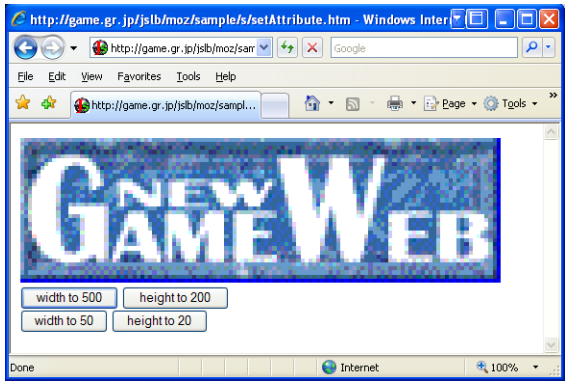
File Edit Format View Help

```
<HTML><HEAD><TITLE></TITLE></HEAD><BODY>
<FORM name=f0>
<INPUT onclick="alert(this.getAttribute('type'))" type=button value=type name=e0>
<INPUT onclick="alert(this.getAttribute('name'))" type=button value=name name=e1>
<INPUT onclick="alert(this.getAttribute('value'))" type=button value=value name=e2>
<INPUT onclick="alert(this.getAttribute('onClick'))" type=button value=onClick name=e3>
</FORM></BODY></HTML>
```

Example 13: DOM and Setting Attributes



initial



Change Width to 500

Change Height to 200

setAttribute[1] - Notepad

```
<HTML><HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
<!--
  function getOj(id){
    return document.getElementById( id ); }
//--> </SCRIPT></HEAD><BODY>
<IMG ID="imgTest"
     SRC="http://game.gr.jp/GameWeb/NGWtools/images/logo01.gif"><BR>
<INPUT TYPE=button
       VALUE="width to 500"
       onClick="getOj('imgTest').setAttribute('width', 500 )">
<INPUT TYPE=button
       VALUE="height to 200"
       onClick="getOj('imgTest').setAttribute('height', 200 )"><BR>
<INPUT TYPE=button
       VALUE="width to 50"
       onClick="getOj('imgTest').setAttribute('width', 50 )">
<INPUT TYPE=button
       VALUE="height to 20"
       onClick="getOj('imgTest').setAttribute('height', 20 )">
</BODY></HTML>
```

Nodes a DOM Can Contain

- An Example

```
<sentence>   The &projectName; <![CDATA[<i>project</i>]]>
   is   <?editor: red><bold>important</bold><?editor: normal>.
   </sentence>
```

  - contains an entity ref., CDATA section, processing instructions (<?...?>

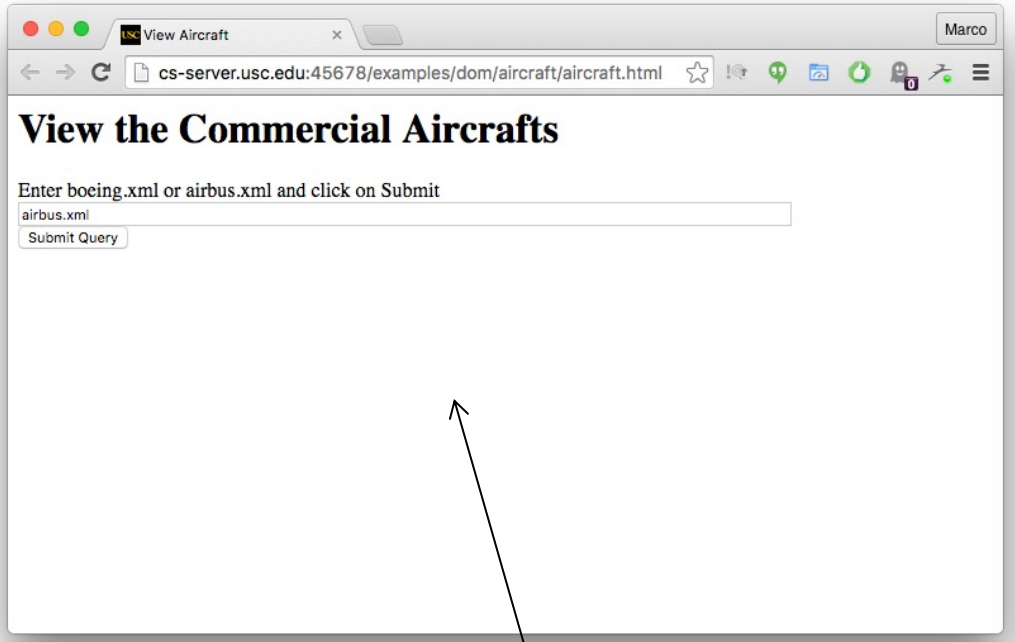
- Its DOM structure looks like this:

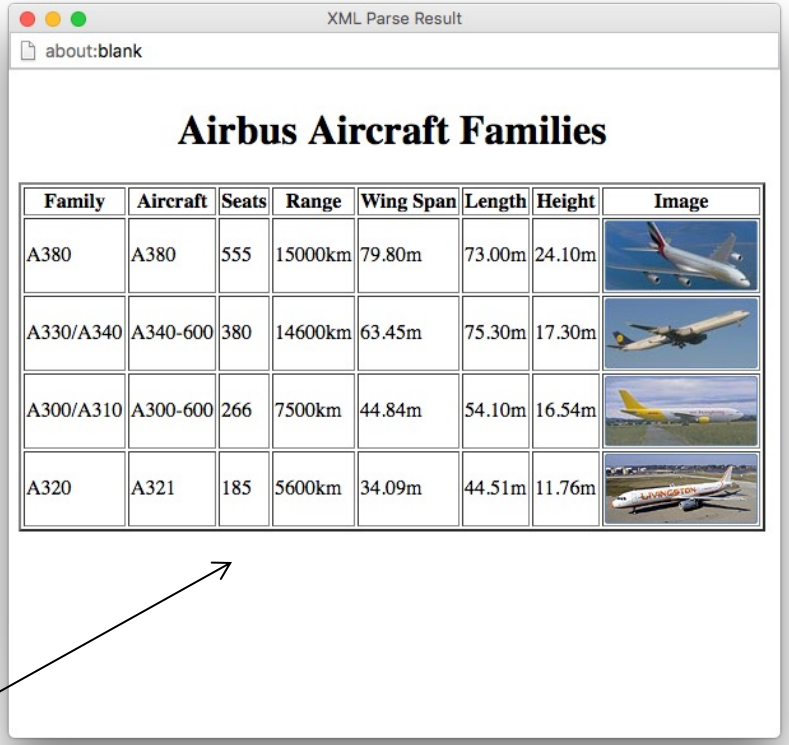
```
+ ELEMENT: sentence
   + TEXT: The
   + ENTITY REF: projectName
     + COMMENT: The latest name we're using
     + TEXT: Eagle
   + CDATA: <i>project</i>
   + TEXT: is
   + PI: editor: red
   + ELEMENT: bold
     + TEXT: important
   + PI: editor: normal
```

Summary of XML/HTML node types and children

- Document -- Element(maximum of one), ProcessingInstruction, Comment, DocumentType (maximum of one)
- DocumentFragment -- Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
- DocumentType -- no children
- EntityReference -- Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
- Element -- Element, Text, Comment, ProcessingInstruction, CDATASection, EntityReference
- Attr -- Text, EntityReference
- ProcessingInstruction -- no children
- Comment -- no children
- Text -- no children
- CDATASection -- no children
- Entity -- Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
- Notation -- no children

Example 14: A Longer DOM Example

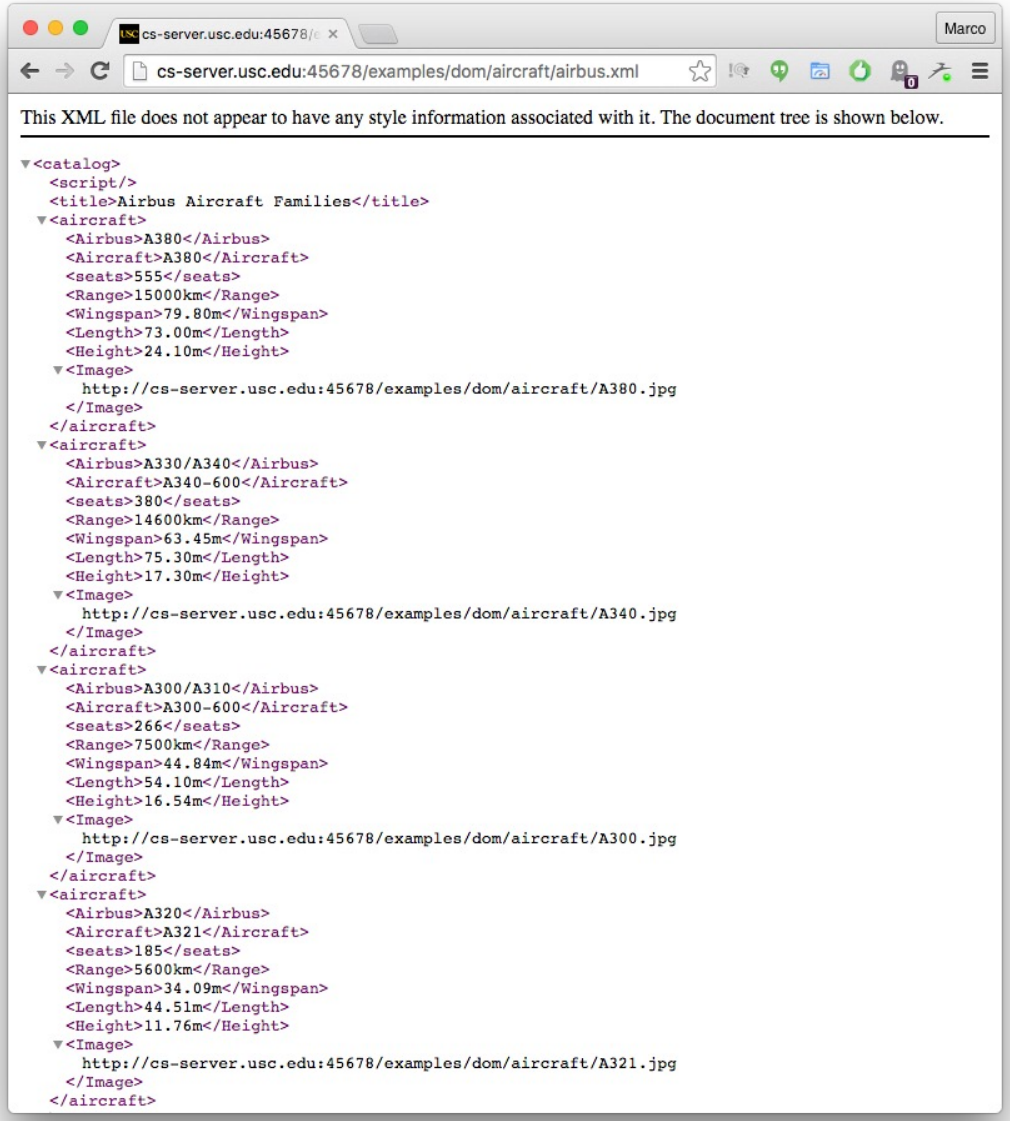




Given a URL of an XML file that describes a set of aircraft, re-format the data into an HTML page

See: https://csci571.com/examples/dom/aircraft/aircraft.html

airbus.xml



HTML Code for the Initial Input

```
<h1>View the Commercial Aircrafts </h1>
Enter XML file

<form name="myform" method="POST" id="location">
  <input type="text" name="URL" maxlength="255"
         size="100" value="airbus.xml" />
  <br />
  <input type="button" name="submit"
         value="Submit Query"
         onClick="viewXML(this.form)" />
</form>
```

viewXML Routine

```
function viewXML(what) {
  var URL = what.URL.value;
  function loadXML(url) {
    if (window.XMLHttpRequest)
        xmlhttp = new XMLHttpRequest(); // code for IE7+, Firefox, Chrome, Opera, Safari
    else
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP"); // code for IE6, IE5
    xmlhttp.open("GET", url, false);
    xmlhttp.send();
    xmlDoc = xmlhttp.responseXML;
    return xmlDoc;
  }
  xmlDoc = loadXML(URL);
  if (window.ActiveXObject) {
    //if IE, simply execute script (due to async prop).
    if (xmlDoc.parseError.errorCode != 0) {
       var myErr = xmlDoc.parseError;
       generateError(xmlDoc);
       hWin = window.open("", "Error", "height=300,width=340");
       hWin.document.write(html_text);
    } else {
       generateHTML(xmlDoc);
       hWin = window.open("", "Assignment4", "height=800,width=600");
       hWin.document.write(html_text);
    }
  } else { //else if FF, execute script once XML object has loaded
    xmlDoc.onload = generateHTML(xmlDoc);
    hWin = window.open("", "Assignment4", "height=800,width=600");
    hWin.document.write(html_text);
  }
  hWin.document.close();
}
```

generateXML Function

```
function generateHTML(xmlDoc) {
  ELEMENT_NODE = 1; // MS parser doesn't define Node.ELEMENT_NODE
  root = xmlDoc.DocumentElement;
  const caption = xmlDoc.getElementsByTagName("title").item(0).firstChild.nodeValue;

  html_text = `
    <html><head><title>XML Parse Result</title></head><body>
      <table border='2'>
      <caption align='left'><h1>${caption}</h1></caption>";
      <tbody><tr>`;
  x = 0;
  y = 0;
  const planes = xmlDoc.getElementsByTagName("aircraft");
  planeNodeList = planes.item(0).childNodes;
  // output the headers
  for (let planeNodeChild of planeNodeList) {
    if (planeNodeChild.nodeType != ELEMENT_NODE) continue;
    header = planeNodeChild.nodeName;
    if (header == "Airbus") { header = "Family"; x = 120; y = 55; }
    if (header == "Boeing") { header = "Family"; x = 100; y = 67; }
    if (header == "seats") header = "Seats";
    if (header == "Wingspan") header = "Wing Span";
    if (header == "height") header = "Height";
    html_text += "<th>" + header + "</th>";
  }
  html_text += "</tr>";
```

generateXML Function (cont'd)

```
// output out the values
for (let plane of planes) {
  html_text += "<tr>"; //start a new row of the output table
  for (let planeNodeChild of plane.childNodes) { //get properties of a plane
    if (planeNodeChild.nodeType != ELEMENT_NODE) continue;
    if (planeNodeChild.nodeName == "Image") {
      html_text += `
        <td>
          <img src='${planeNodeChild.firstChild.nodeValue}' width='${x}' height='${y}'>
        </td>`;
    } else {
      html_text += "<td>" + planeNodeChild.firstChild.nodeValue + "</td>";
    }
  }
  html_text += "</tr>";
}

html_text += "</tbody> </table> </body> </html>";
}
```

Example 15: Another DOM Example
A simple XML file for a bookstore

```
−<bookstore>
  −<book category="cooking">
      <title lang="en">Everyday Italian</title>
      <author>Giada De Laurentiis</author>
      <year>2005</year>
      <price>30.00</price>
    </book>
  +<book category="children"></book>
  +<book category="web"></book>
  +<book category="web" cover="paperback"></book>
</bookstore>
```


Example Cont'd – function xmlparse traverses and outputs the books

```
function displayString(out) {
  document.getElementById("output").innerHTML = out;
}
function xmlparse() {
  var html = "";
  xmlDoc = loadXML("bookstore.xml");
  html += "xmlDoc is loaded, ready for use<br />";
  var bookstore = xmlDoc.documentElement;
  for (const book of bookstore.childNodes) {
    if (book.nodeType !== 1) continue
    html += "<ul><li>";
    html += "<b>" + book.nodeName + " : </b>";
    for (const child of book.childNodes) {
      if (child.nodeType !== 1) continue
      html += child.nodeName + ": " +                    //-> title, author etc
              child.childNodes[0].nodeValue + "<br />"; //-> text values
    }
    html += "</li></ul>";
  }
  displayString(html);
}
</script></head><body><h2>This is the domtest web page</h2>
<input type="button" name="submit" value="Submit Query“ onClick="xmlparse()" />
<noscript><div id="output"></div></body></html>
```

Before and After

An alternate solution that makes use of "bookstore.children" instead of Childnodes can be found at http://csci571.com/examples/dom/example2.html (Example 17)

Additional Reads

DOM Examples at w3schools.com:

https://www.w3schools.com/jsref/dom_obj_document.asp

See **Document Object Properties and Methods** section

(hint: try all examples])

https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model
https://developer.mozilla.org/en-US/docs/Web/API/Node
https://developer.mozilla.org/en-US/docs/Web/API/Element