

CIS 4004: Web Based Information Technology Fall 2012

JavaScript – Part 2

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
<http://www.cs.ucf.edu/courses/cis4004/fall2012>

Department of Electrical Engineering and Computer Science
University of Central Florida



The Document Object Model

- The Document Object Model (DOM) for HTML5 represents a hierarchy tree.
- At the root of every web page or document is the `<html>` element, and the rest of the elements in the page are a branch somewhere along the tree.
- JavaScript uses the DOM for addressing and manipulation a web page beyond what you can do with HTML5 alone.
- The entire DOM tree is a representation of the document that resides in your computer's memory.



The Document Object Model

- When any part of the DOM tree is addressed, it does so by referencing an element within the tree, beginning with `document`.
- Each element in the tree is addressed in order of the hierarchy beginning with `document`.
- The different elements in a web page are the different properties or methods (built-in functions) of the `document` separated by a dot (`.`).



The Document Object Model

- For example, `document.forms.signup;` would address a form named `signup` within a document.
- A built-in function that does something with the document would appear as:

```
document.write("this is some text.");
```

- The `window` root along with the document has several built-in functions that are useful for manipulating viewing areas of a web page (more later). See the example markup on page 15 in the JavaScript – Part 1 notes to see a use of the `window` root.



Viewing A Document's DOM Tree

- Current browsers provide developer tools that can display a visual representation of a document's DOM tree.
- The table on the next page illustrates how to access the developer tools for desktop versions of each of the major browsers.
- For the most part, the developer tools are very similar across the browsers.
- NOTE: For FireFox, you must first install the DOM Inspector add-on available at: [https://addons/mozilla/org/en-US/firefox/addon/dom-inspector-6622/](https://addons.mozilla.org/en-US/firefox/addon/dom-inspector-6622/).



Viewing A Document's DOM Tree

Browser	Command to display developer tools
Chrome	Windows/Linux: <i>Control + Shift + i</i> Mac OS X: <i>Command + Option + i</i>
Firefox	Windows/Linux: <i>Control + Shift + i</i> Mac OS X: <i>Command + Option + i</i>
IE	<i>F12</i>
Opera	Windows/Linux/Mac OS X: From View on tool bar select Developer Tools then select Opera DragonFly (Control + Shift + i should also work)
Safari	Windows/Linux/Mac OS X: From Edit/Preferences/Advanced check "Show develop menu in menu bar" - then select as needed



Viewing A Document's DOM Tree

- We'll use the markup shown on the next page as an example to view the DOM tree in a couple of the browsers so that you can see what the developer tool looks like.
- The tool in Chrome is shown on page 9, the tool in IE is shown on page 10, the tool in Safari is shown on page 11, and the tool in Opera is shown on page 12.





```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="utf-8">
5     <title>Basic DOM Tree Demonstration</title>
6 </head>
7 <body>
8     <h1>An HTML5 Page</h1>
9     <p>This page contains some basic HTML5 elements. The DOM tree
10     for the document contains a DOM node for every element</p>
11     <p>Here's an unordered list:</p>
12     <ul>
13         <li>One</li>
14         <li>Two</li>
15         <li>Three</li>
16     </ul>
17 </body>
18 </html>
19
```



Basic DOM Tree Demonstr: x

file:///K:/CIS%204004%20-%20Fall%202012/code/JavaScript%20-%20Part%202/domtree.html

Free Hotmail | KeepVid- Download ... | Suggested Sites | Web Slice Gallery | Other bookmarks

An HTML5 Page

This page contains some basic HTML5 elements. The DOM tree for the document contains a DOM node for every element

Here's an unordered list:

- One
- Two
- Three

Elements | Resources | Network | Sources | Timeline | Profiles | Audits | Console

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Basic DOM Tree Demonstration</title>
  </head>
  <body>
    <h1>An HTML5 Page</h1>
    <p>
      "This page contains some basic HTML5 elements. The DOM tree
      for the document contains a DOM node for every element"
    </p>
    <p>Here's an unordered list:</p>
    <ul>
      <li>One</li>
      <li>Two</li>
      <li>Three</li>
    </ul>
  </body>
</html>
```

Computed Style

Styles

```
element.style {
}

Matched CSS Rules
body {
  display: block;
  margin: 8px;
}

Metrics
Properties
DOM Breakpoints
Event Listeners
```



Basic DOM Tree Demonstration - Developer Tools

File Find Disable View Outline Images Cache Tools Validate | Browser Mode: IE8 Document Mode: IE8 Standards

HTML CSS Script Profiler

Style Trace Styles **Layout** Attributes

```

<!-- DOCTYPE html -->
<html lang="en">
  <head>
    <title>Basic DOM Tree Demonstration</title>
    <meta charset="utf-8"/>
  <body>
    <h1>
      Text - An HTML5 Page
    <p>
      Text - This page contains some basic HTML5 elements. The DO
    <p>
      Text - Here's an unordered List:
    <ul>
      <li>
        Text - One
      <li>
        Text - Two
      <li>
        Text - Three
  </body>
</html>

```

Offset: 0

Margin: 8px

Border: medium

Padding: 0px

Content: 926x184

Coordinates: (0, 0) Z-index = auto



Basic DOM Tree Demonstration

File Edit View History Bookmarks Develop Window Help

file:///K:/CIS%204004%20-%20Fall%202012/code/JavaScript Google

Apple Yahoo! Google Maps YouTube Wikipedia News (105) Popular

An HTML5 Page

This page contains some basic HTML5 elements. The DOM tree for the document contains a DOM node for every element

Here's an unordered list:

- One

Elements Resources Network Scripts Timeline Search Elements

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Basic DOM Tree Demonstration</title>
  </head>
  <body>
    <h1>An HTML5 Page</h1>
    <p>
      "This page contains some basic HTML5
      elements. The DOM tree
      for the document contains a
      DOM node for every element"
    </p>
    <p>Here's an unordered list:</p>
    <ul>
      <li>One</li>
      <li>Two</li>
      <li>Three</li>
    </ul>
  </body>
</html>

```

Computed Style Show inherited

Styles

```

element.style {
}

```

Matched CSS Rules

```

body {
  display: block;
  margin: 8px;
}

```

Metrics Properties Event Listeners

html body



Basic DOM Tree Demonstration - Opera

File Edit View Bookmarks Tools Help

Speed Dial Basic DOM Tree Demo... +

Local localhost/K:/CIS%204004%20-%20Fall%202012/code/J Search with Google

An HTML5 Page

This page contains some basic HTML5 elements. The DOM tree for the document contains a DOM node for every element

Here's an unordered list:

- One
- Two
- Three

Basic DOM Tree Demonstration

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8"/>
    <title>Basic DOM Tree Demonstration</title>
  </head>
  <body>
    <h1>An HTML5 Page</h1>
    <p>This page contains some basic HTML5 elements.
    The DOM tree for the document contains a DOM
    node for every element</p>
    <p>Here's an unordered list:</p>
    <ul>
      <li>One</li>
      <li>Two</li>
      <li>Three</li>
    </ul>
  </body>
</html>

```

Styles Properties Layout Search

Filter

Computed Style

Styles

```

body {
  display: block;
  margin: 8px;
}

```

New Style

html body



Viewing A Document's DOM Tree

- Let's focus for a bit on the tool as it appears in the Chrome browser.
- A node in the DOM tree can be expanded and collapsed using the ► and ▼ arrows next to a given node. The screen shot on page 9 illustrates all nodes in the document fully expanded.
- The `html` node is the root of the tree since it has no parent. Notice in the screen shot on the next page, that if the cursor is placed on the `html` node, the entire document is highlighted in the top window.



Basic DOM Tree Demonstration - Opera

File Edit View Bookmarks Tools Help

Speed Dial Basic DOM Tree Demo...

Local localhost/K:/CIS%204004%20-%20Fall%202012/code/J

An HTML5 Page

This page contains some basic HTML5 elements. The DOM tree for the document contains a DOM node for every element

Here's an unordered list:

- One
- Two
- Three

Basic DOM Tree Demonstration

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8"/>
    <title>Basic DOM Tree Demonstration</title>
  </head>
  <body>
    <h1>An HTML5 Page</h1>
    <p>This page contains some basic HTML5 elements.
    The DOM tree for the document contains a DOM
    node for every element</p>
    <p>Here's an unordered list:</p>
    <ul>
      <li>One</li>
      <li>Two</li>
      <li>Three</li>
    </ul>
  </body>
</html>

```

Styles Properties Layout Search

Filter

- accessKey ""
- all HTMLCollection
- attributes NamedNodeMap
- baseURI "file:///localhost/K:/CIS%204004_..."
- childElementCount 2
- childNodes NodeList
- children HTMLCollection
- classList DOMTokenList
- className ""
- clientHeight 145
- clientLeft 0
- clientTop 0
- clientWidth 678
- contentEditable "inherit"
- currentPage 0
- currentStyle CSSStyleDeclaration
- dataset DOMStringMap



Viewing A Document's DOM Tree

- When you select a node in the left side of the developer's tools Elements tab, the node's details are displayed in the right side.
- On the next page, I've illustrated this by selecting the `<p>` element just before the start of the unordered list. In the Properties section (right pane) you can see the values for the `<p>` element.



An HTML5 Page

This page contains some basic HTML5 elements. The DOM tree for the document contains a DOM node for every element

Here's an unordered list:

- One
- Two
- Three

Elements Resources Network Sources Timeline Profiles Audits Console

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Basic DOM Tree Demonstration</title>
  </head>
  <body>
    <h1>An HTML5 Page</h1>
    <p>
      "This page contains some basic HTML5 elements. The DOM tree
      for the document contains a DOM node for every element"
    </p>
    <p>Here's an unordered list:</p>
    <ul>
      <li>One</li>
      <li>Two</li>
      <li>Three</li>
    </ul>
  </body>
</html>
```

Properties

- HTMLParagraphElement
 - accessKey: ""
 - align: ""
 - attributes: NamedNodeMap
 - baseURI: "file:///K:/CIS%204004%20-%20..."
 - childElementCount: 0
 - childNodes: NodeList[1]
 - children: HTMLCollection[0]
 - classList: DOMTokenList
 - className: ""
 - clientHeight: 20
 - clientLeft: 0
 - clientTop: 0
 - clientWidth: 828
 - contentEditable: "inherit"
 - dataset: DOMStringMap
 - dir: ""
 - draggable: false
 - firstChild: Text



Viewing A Document's DOM Tree

- In addition to viewing a document's DOM structure, the developer tools in each browser typically allow you to view and modify styles, view and debug JavaScripts used in the document, view the resources (such as images) used by the document, and so on.
- I would suggest that you become familiar with the developer tool in whichever browser you intend to use as your primary development environment.



The Document Object Model

- To get a better sense of how the DOM works with your web page and JavaScript, it helps to see what can be done with a web page's windows – the viewing part of your web page.
- The following example shows how to load a new window from the current document, leaving the current page in place.



PageOpener.html

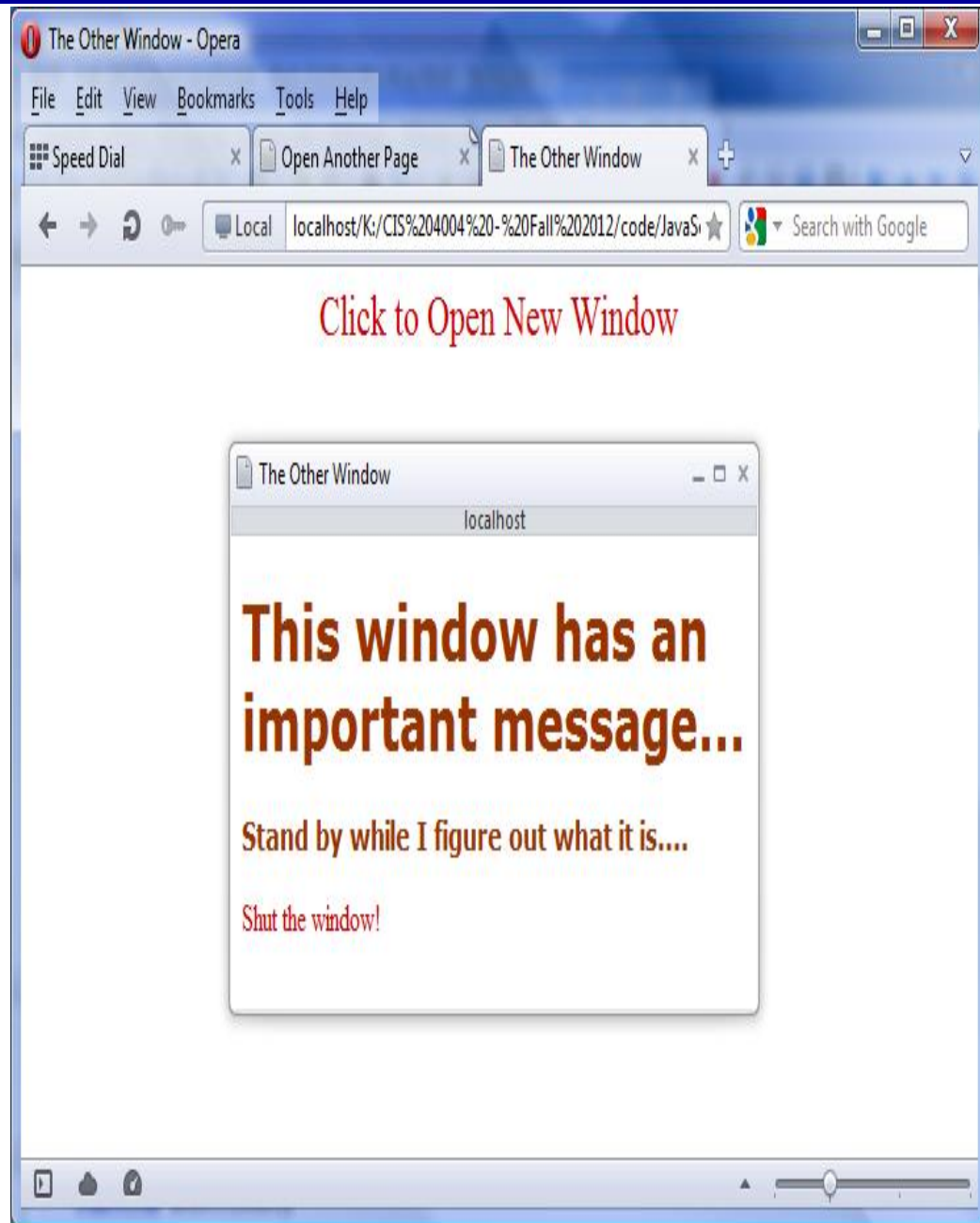
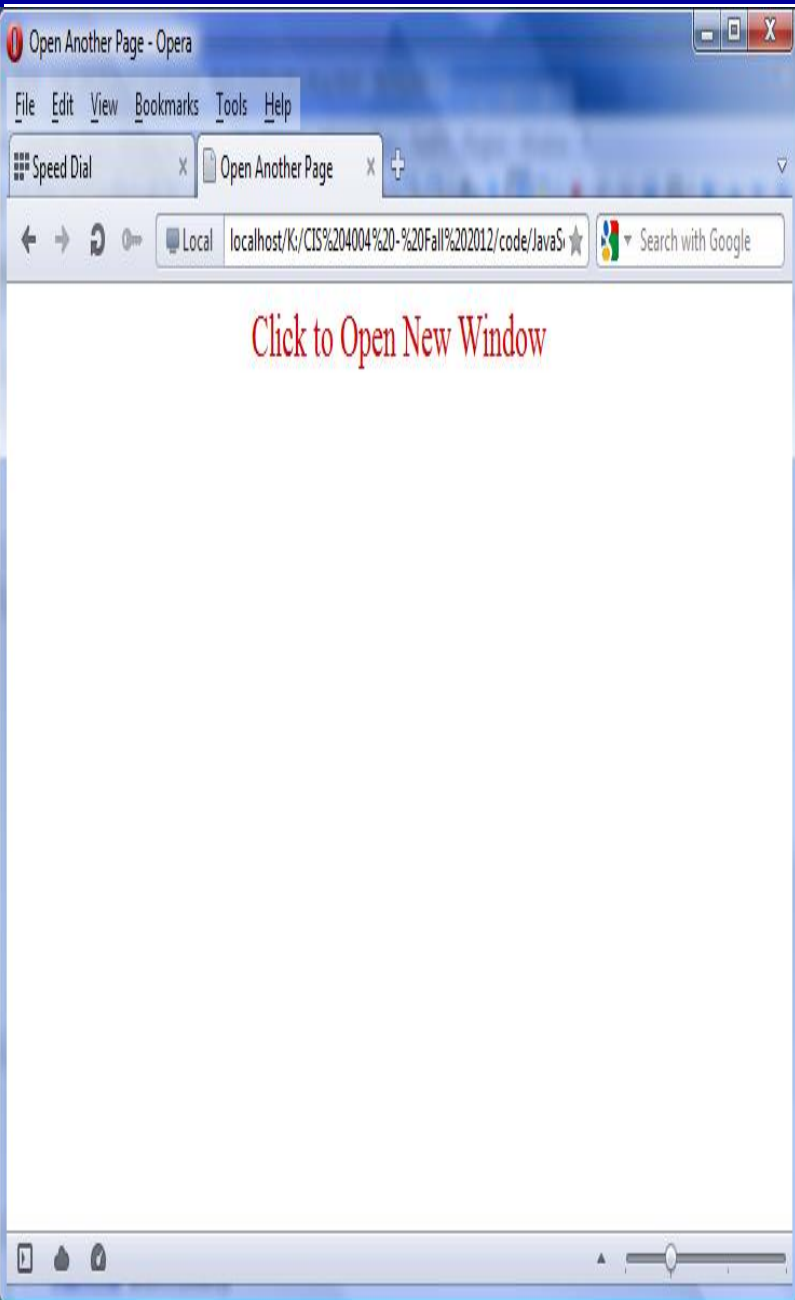
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <title> Open Another Page </title>
5   <meta charset="utf-8">
6   <style type="text/css">
7     <!--
8       a{
9         text-decoration:none;
10        color:#cc0000;
11        font-size:24px;
12      }
13      header{
14        text-align:center;
15      }
16    -->
17  </style>
18 </head>
19 <body>
20   <header> <a href="#" onClick="someOtherWindow()">Click to Open New Window</a> </header>
21   <script type="text/javascript">
22     function someOtherWindow()
23     {
24       window.open("OtherWindow.html","ow","width=400,height=200");
25     }
26   </script>
27 </body>
28 </html>
```



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <title>The Other Window </title>
5   <meta charset="utf-8">
6   <style type="text/css">
7     <!--
8       h1,h4{
9         font-family:Verdana, Geneva, sans-serif; color:#930;
10      }
11      a{
12        text-decoration:none; color:#cc0000;text-align:center;
13      }
14     -->
15   </style>
16 </head>
17 <body>
18   <h1>This window has an important message...</h1>
19   <h4>Stand by while I figure out what it is....</h4>
20   <a href="#" onClick="shutItDown()">Shut the window!</a>
21   <script type="text/javascript">
22     function shutItDown()
23     {
24       window.close();
25     }
26   </script>
27 </body>
28 </html>
```

OtherWindow.html





The Document Object Model

- Up to this point when we've written markup where one page is linked to another page, the current page has disappeared as soon as the user clicks the link to the other page.
- Now, using this little bit of JavaScript you can “talk” directly to the page and tell it you want a new window of a specified size to open while the current window stays open.



HTML5 Elements And The DOM

- In order to give you an even better idea of how to work with the DOM in HTML5, certain new elements require DOM references within the tags themselves.
- One such new element is the `<output>` element. Currently, Opera is the only browser that has fully implemented this element, so again, you should test the following markup using Opera.
- When you use the `<output>` element, you can place the results of a calculation directly on the webpage. You don't have to build a JavaScript function or even a simple script.



HTML5 Elements And The DOM

- However, the materials within an output element must follow the same DOM rules as with JavaScript.
- The output container doesn't require content between the opening and closing tags. However, all of the calculations must be within the `<output>` element itself.
- The `<output>` element works in conjunction with the `<form>` element and we've already covered that and your current project deals with that as well.
- Now we want to focus on the DOM structure in the `<output>` element's use. Consider the following markup.




```
*K:\CIS 4004 - Fall 2012\code\JavaScript - Part 2\shoppingOutput.html - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
PageOpener.html OtherWindow.html dom.html shoppingOutput.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <title> A Simple Shopping Cart Cost Calculator</title>
6   <style type="text/css">
7     <!--
8       /*042B45,FFC54F,FFE6BF,E8A5B5,FF0A03*/
9       body { font-family:Verdana, Geneva, sans-serif; background-color:#FFE6BF; color:#042B45;}
10      input { background-color:#FFE6BF;}
11      h1 { color:#E8A5B5; background-color:#042B45; text-align:center; }
12      h3 { color:#FFC54F; background-color:#FF0A03; }
13    -->
14  </style>
15 </head>
16 <body>
17   <header>
18     <h1>Shopping Calculator</h1>
19   </header>
20   <form>
21     <input name=cost type=number> &nbsp;&nbsp;&nbsp;Cost <br>
22     <input name=tax type=number> &nbsp;&nbsp;&nbsp;Tax--Enter as decimal percent (e.g., .06) <br>
23     <h3> &nbsp;&nbsp;&nbsp;Total = $
24       <output onforminput="value = cost.valueAsNumber * tax.valueAsNumber + cost.valueAsNumber"></output>
25     </h3>
26   </form>
27 </body>
28 </html>
```

Hyper Text Markup Language file length : 827 lines : 29 Ln : 22 Col : 35 Sel : 0 UNIX ANSI as UTF-8 INS



A Simple Shopping Cart Cost Calculator - Opera

File Edit View Bookmarks Tools Help

Speed Dial x A Simple Shopp... x A Simple Shopp... x A Simple Shopp... x A Simple Shopp... x

Local localhost/K:/CIS%204004%20-%20Fall%202012/code/J Search with Google

Shopping Cart Cost Calculator

Cost

Tax--Enter as decimal percent (e.g., .06)

Total = \$ 107.06

This output is produced via the `onforminput` event handler.



Analysis Of The Previous Example

- Within the form container, two input elements are named `cost` and `tax`. In the context of the DOM, each is an object with certain properties, one of which is `valueAsNumber`. This is illustrated by the screen shot on the next page.
- Whatever number character is in the input form is treated as an actual number instead of a text character. The `valueAsNumber` is a property of the `<input>` element and not the `number` type that was used in the element. (I could have used a `text` value for the input type and had the same results using the input element.) Recall that the `number` type simply provides the selection list (see Inside HTML5 – Part 4 – Forms, pages 56-58 for more details.)



Shopping Cart Cost Calculator

Cost

Tax--Enter as decimal percent (e.g., .06)

```

<form>
  <input name="cost" type="number"/>
  Cost
  <br/>
  <input name="tax" type="number"/>
  Tax--Enter as decimal percent (e.g., .06)
  <br/>
  <h3>
    Total = $
    <output id="value" onforminput="value =
      cost.valueAsNumber * tax.valueAsNumber +
      cost.valueAsNumber"/>
  </h3>
</form>
</body>

```

Filter

- type "number"
- unselectable ""
- useMap ""
- validationMessage ""
- validity ValidityState
- value ""
- valueAsDate null
- valueAsNumber NaN
- width 0
- willValidate true
- HTMLInputElementPrototype
- HTMLElementPrototype
- ElementPrototype



Analysis Of The Previous Example

- The number type simply provides the “spinner” input control (the up/down arrows) window, but values in the input window are not automatically converted into numeric data.
- Notice how the `onforminput` event handler works. As information is entered into the form, the results are calculated and displayed.
- After the user has entered the cost, but before they have entered the tax, the result will be displayed as NaN (Not a Number) because the tax value is null, resulting in a non-numeric result. However, as soon as the tax is entered, the output changes to a number. See the next two screen shots.



A Simple Shopping Cart Cost Calculator - Opera

File Edit View Bookmarks Tools Help

Speed Dial x A Simple Shopp... x A Simple Shopp... x A Simple Shopp... x A Simple Shopp... x

Local localhost/K:/CIS%204004%20-%20Fall%202012/code/J Search with Google

Shopping Cart Cost Calculator

Cost

Tax--Enter as decimal percent (e.g., .06)

Total = \$ NaN

The user has only entered the cost amount. At this point the tax value is null so the total is not a numeric value and NaN is written as the total.



A Simple Shopping Cart Cost Calculator - Opera

File Edit View Bookmarks Tools Help

Speed Dial x A Simple Shopp... x A Simple Shopp... x A Simple Shopp... x A Simple Shopp... x

Local localhost/K:/CIS%204004%20-%20Fall%202012/code/J Search with Google

Shopping Cart Cost Calculator

Cost

Tax--Enter as decimal percent (e.g., .06)

Total = \$ 107.06

The user has now entered both numbers into the form and the total is correctly calculated and displayed.



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Temperature Converter Using JavaScript Functions</title>
  <style type="text/css">
    <!--
      body { background-color:gray; color:blue; font-size:16pt; font-
weight:bold;}
      h2 { color:red;}
      p { color:yellow; font-size:24pt; }
    -->
  </style>
</head>
<body>
  <h2>Temperature Conversion</h2>
  <form>
    <label for="temperature">Temperature:</label>
    <input type="text" id="temperature" />
    <input type="button" id="f_to_c" name="f_to_c" value="F to C" />
    <input type="button" id="c_to_f" name="c_to_f" value="C to F" />
  </form>
  <p id="result"></p>
  <script src="scripts/temperature.js"></script>
</body>
</html>
```




```
var report = function (celsius, fahrenheit) {  
    document.getElementById("result").innerHTML = celsius + "\xb0C = " +  
    fahrenheit + "\xb0F";  
};  
  
document.getElementById("f_to_c").onclick = function () {  
    var f = document.getElementById("temperature").value;  
    report((f - 32) / 1.8, f);  
};  
  
document.getElementById("c_to_f").onclick = function () {  
    var c = document.getElementById("temperature").value;  
    report(c, 1.8 * c + 32);  
};
```



Temperature Converter Using JavaScript Functions

Temperature Convert... x +

Local localhost/Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/JavaScript%20-%20 Search with Google

Temperature Conversion

Temperature:

40°C = 104°F



Temperature Converter Using JavaScript Functions

Temperature Convert... x

Local localhost/Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/JavaScript%20-%20 Search with Google

Temperature Conversion

Temperature:

0°C = 32°F



K:\CIS 4004 - Fall 2012\code\JavaScript - Part 2\SimpleVariable.html - Notepad++

File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?

PageOpener.html OtherWindow.html dom.html shoppingOutput.html UserObject.html SimpleVariable.html

```
12 </head>
13 <body>
14 <script type="text/javascript">
15 function advertisement() {
16     markVar="Brought to you by Mark's variable.";
17     return markVar;
18 }
19 //Variable with function
20 popUpAd=advertisement();
21 document.write(popUpAd);
22 //Variable with HTML5 code
23 cr="<br>";
24 document.write(cr);
25 // Variable with string
26 funHouse=" Elm Street";
27 // Boolean variable
28 var fate=true;
29 // Variable with string
30 query="Will I find true happiness in HTML5";
31 // Variables with numbers
32 fun=100;
33 house=23;
34 // Math with variables
35 funPlusHouse=fun + house;
36 // Adding numeric and string variable (concatenation)
37 showAddress=funPlusHouse + funHouse;
38
39 browser=navigator.platform;
```

Hyper Text Markup Language file length: 1141 lines: 49

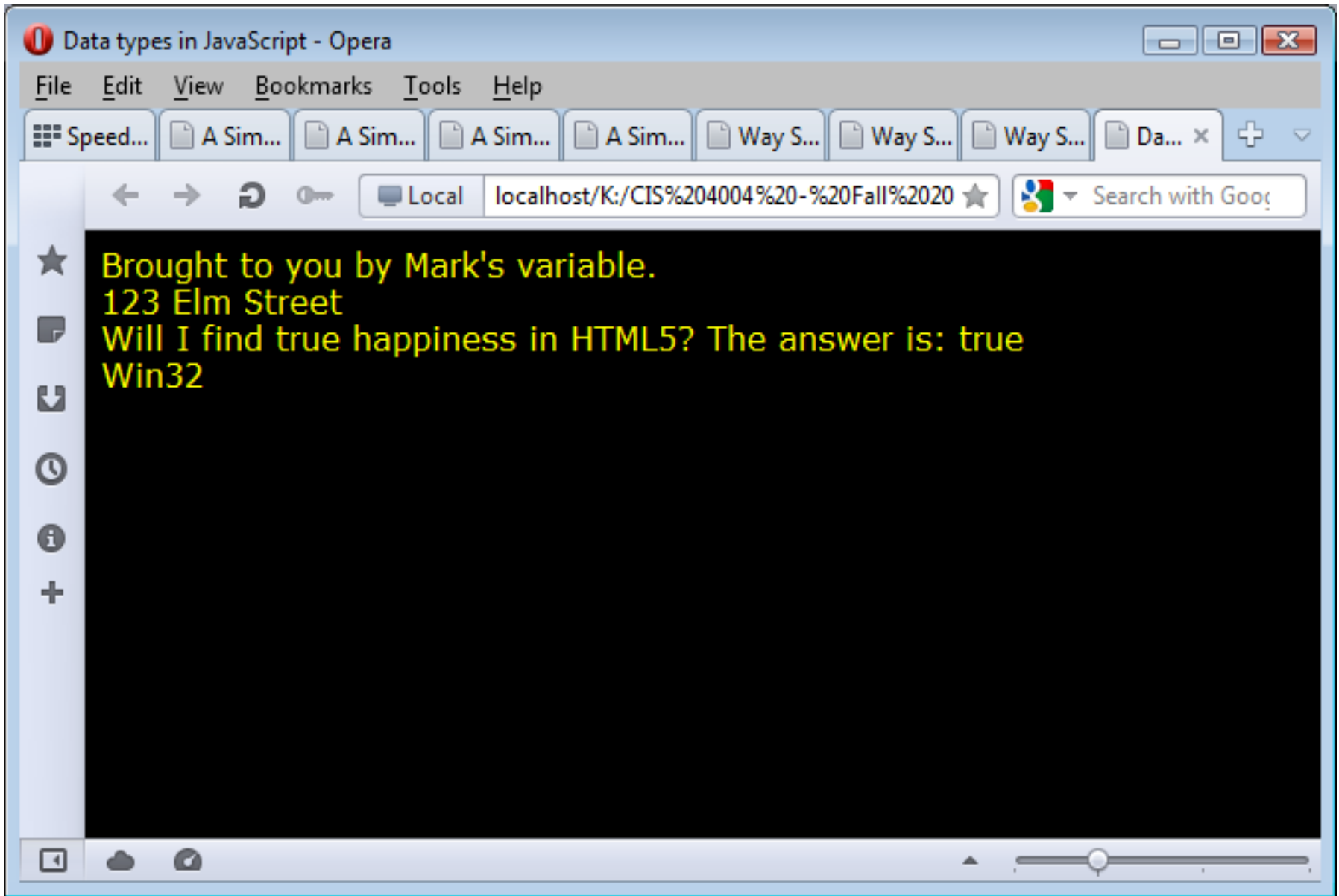
K:\CIS 4004 - Fall 2012\code\JavaScript - Part 2\SimpleVariable.html - Note...

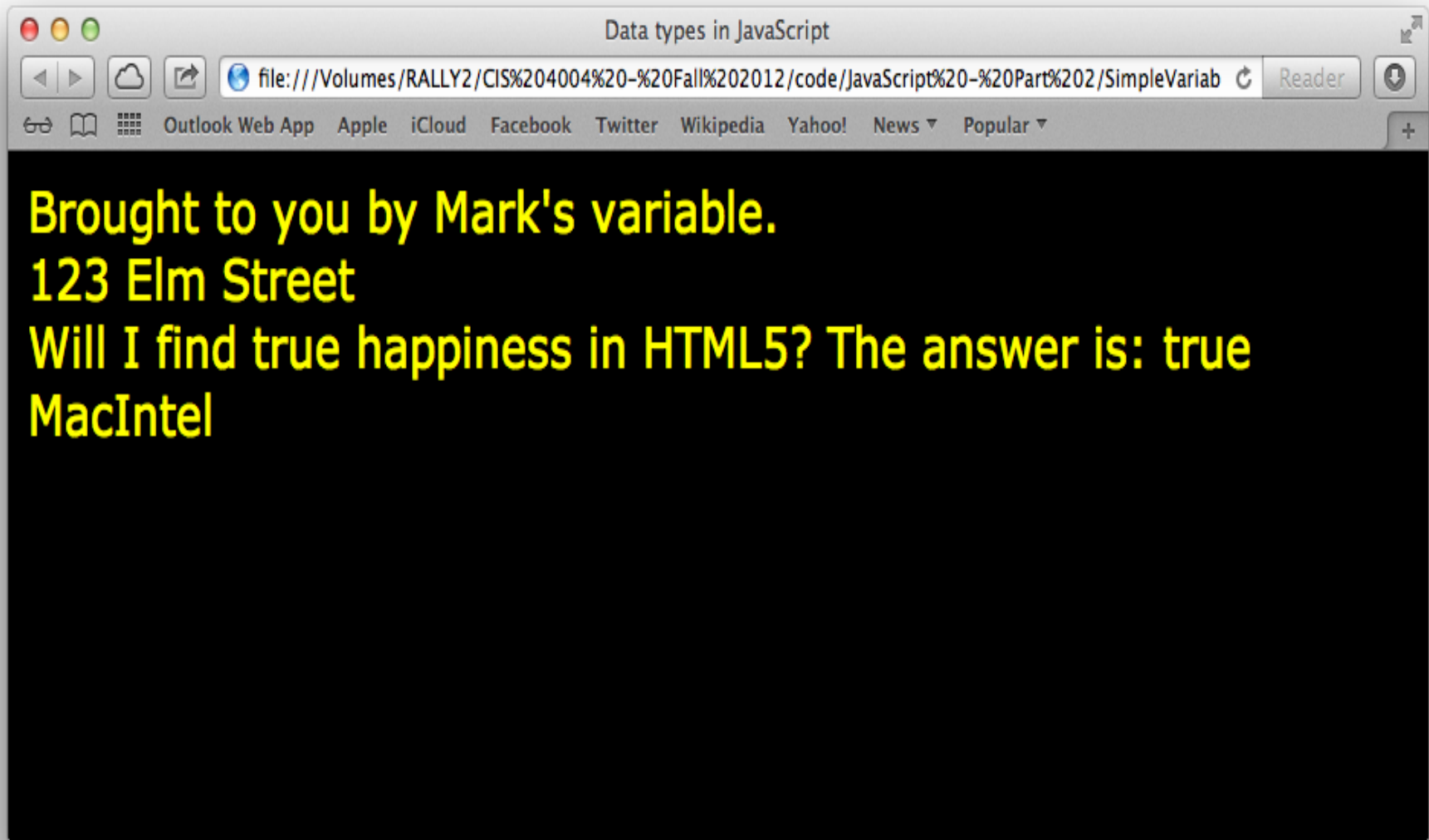
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?

```
33 house=23;
34 // Math with variables
35 funPlusHouse=fun + house;
36 // Adding numeric and string variable (concatenation)
37 showAddress=funPlusHouse + funHouse;
38
39 browser=navigator.platform;
40 document.write(showAddress);
41 document.write(cr);
42 document.write(query);
43 document.write(fate);
44 document.write(cr);
45 document.write(browser);
46 </script>
47 </body>
48 </html>
```

Ln: 21 Col: 29 Sel: 0 UNIX ANSI INS







Traversing And Modifying A DOM Tree

- The DOM enables you to programmatically access a document's elements, allowing you to modify its contents dynamically using JavaScript.
- The HTML5/CSS/JavaScript example we'll use is available on the course website, I did not include the markup in these notes. The example will allow you to traverse the DOM tree, modify nodes and create or delete content dynamically.
- The CSS class `highlighted` is applied dynamically to elements in the document as they are selected, added, or deleted using the form at the bottom of the document.
- As you play around with this example, be sure to do it in the developer tool so that you can see the DOM tree as well.



Traversing And Modifying A DOM Tree

- The HTML5 document is manipulated dynamically by modifying its DOM tree.
- Each element has an `id` attribute, which is also displayed in square brackets at the beginning of the element (so you can see which element is which).
- The `click` event listeners are registered in the JavaScript (available on the course website) for the six buttons that call corresponding functions to perform the actions described by the button's values.



Traversing And Modifying A DOM Tree

- The JavaScript begins by declaring two variables.
- Variable `currentNode` keeps track of the currently highlighted node (the initially highlighted node is the `[bigheading]`), the functionality of each button depends on which node in the document (DOM tree) is currently selected.
- The function `start` registers the event handlers for the buttons, then initializes the `currentNode` to the `<h1>` element, the element with `id = bigheading`.
- Note that the function `start` is called when the window's load event occurs.



Traversing And Modifying A DOM Tree

- The JavaScript variable `idcount` is used to assign a unique id to any new elements that are dynamically created by the user.
- The remainder of the JavaScript contains the event handling functions for the buttons and two helper functions (`switchTo` and `createNewNode`) that are called by the event handlers.
- Over the next few pages, I'll explain how each of the buttons and its corresponding event handler works. Before reading on, you should download the markup, the style sheet, and the JavaScript files and play around with the page a bit to get a feel for what's happening with the page as the user manipulates the page.



Finding and Highlighting an Element Using `getElementById`, `setAttribute` and `getAttribute`

- The first row of the form allows the user to enter the `id` of an element into the text field and click the `Get By id` button to find and highlight the element.
- The button's click event calls function `byId()`.

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

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



Finding and Highlighting an Element Using

`getElementById`, `setAttribute` and `getAttribute`

- First, the `byId()` function uses `getElementById` to assign the contents of the text field to the variable `id`.
- Next, the `byID()` function uses `getElementById` to find the element whose `id` attribute matches the value of variable `id` and assigns this to the variable `target`.
- If an element is found with the specified `id`, and object is returned; otherwise, `null` is returned.
- Next, the function checks to see whether `target` is an object (any object used as a boolean expression is true, while `null` is false). If `target` evaluates to true, the `switchTo()` helper function is called with `target` as its argument.



Finding and Highlighting an Element Using

`getElementById`, `setAttribute` and `getAttribute`

- The `switchTo()` helper function is used a lot in this JavaScript to highlight an element in the page. The current element is given a yellow background (via the CSS class `highlighted`).
- The DOM element methods `setAttribute` and `getAttribute` allow you to modify and get an attribute's value, respectively.
- The function `switchTo` function uses the `setAttribute` method to set the current node's class attribute to the empty string. This clears the class attribute to remove the highlighted class from the `currentNode` before the new node is highlighted.



Finding and Highlighting an Element Using

`getElementById`, `setAttribute` and `getAttribute`

- The last thing the `byID` function does is uses the `getAttribute` method to get the `currentNode`'s `id` and assign it to the input field's `value` property.
- This isn't necessary when this helper function is called by `byID`, but as you'll see later, other functions call `switchTo` as well. In these cases, this line ensures that the text field's `value` contains the currently selected node's `id`.
- Notice that `setAttribute` was not used to change the value of the input field. Methods `setAttribute` and `getAttribute` do not work for user-modifiable content, such as the value displayed in an input field.



Initial Page

[bigheading] HTML5 DOM Tree Demo Page

[smallheading] Element Functionality

[para1] The Document Object Model (DOM) allows for quick, dynamic access to all elements in an HTML5 document for manipulation with JavaScript.

[para2] For more information, check out the "JavaScript lecture notes on the course web site [link] [JavaScript - Part 1, 2, and 3.](#)

[para3] The buttons below demonstrate:(list)

- [item1] getElementById and parentNode
- [item2] insertBefore and appendChild
- [item3] replaceChild and removeChild

bigheading	Get By id
	Insert Before
	Append Child
	Replace Current
Remove Current	
Get Parent	



Initial Page – Shown in Opera DragonFly

[bigheading] HTML5 DOM Tree Demo Page

[smallheading] Element Functionality

[para1] The Document Object Model (DOM) allows for quick, dynamic access to all elements in an HTML5 document for manipulation with JavaScript.

[para2] For more information, check out the "JavaScript lecture notes on the course web site [link]

```

<!DOCTYPE html>
<html lang="en">
  <head>
  <body>
    <h1 id="bigheading" class="highlighted"> [bigheading] HTML5 DOM Tree Demo Page</h1>
    <h3 id="smallheading">
    <p id="para1">
    <p id="para2">
    <p id="para3">
    <ul id="list">
    <div id="nav" class="nav">
  </body>
</html>

```

Filter

- aLink ""
- accessKey ""
- all HTMLCollection
- attributes NamedNodeMap
- background ""
- baseURI "file:///localhost/K:/CIS%204004..."
- bgColor ""
- childElementCount 7
- childNodes NodeList
- children HTMLCollection
- classList DOMTokenList
- className ""
- clientHeight 457
- clientLeft 0
- clientTop 0



User enters "para3" in the text field for the "Get By Id" button. When they click the button the value the user entered into the text field is extracted and the `byId()` function is triggered.

[bigheading] HTML5 DOM Tree Demo Page

[smallheading] Element Functionality

[para1] The Document Object Model (DOM) allows for quick, dynamic access to all elements in an HTML5 document for manipulation with JavaScript.

[para2] For more information, check out the "JavaScript lecture notes on the course web site [\[link\]](#) [JavaScript - Part 1, 2, and 3](#).

[para3] The buttons below demonstrate:(list)

- [item1] `getElementById` and `parentNode`
- [item2] `insertBefore` and `appendChild`
- [item3] `replaceChild` and `removeChild`

para3	Get By id
	Insert Before
	Append Child
	Replace Current
Remove Current	
Get Parent	



User enters "para3" in the text field for the "Get By Id" button. When they click the button the value the user entered into the text field is extracted and the `byId()` function is triggered.

[para2] For more information, check out the "JavaScript lecture notes on the course web site [\[link\]](#) JavaScript - Part 1, 2, and 3.

[para3] The buttons below demonstrate:(list)

- [item1] `getElementById` and `parentNode`
- [item2] `insertBefore` and `appendChild`
- [item3] `replaceChild` and `removeChild`

```

<!DOCTYPE html>
<html lang="en">
  <head>
  <body>
    <h1 id="bigheading" class="highlighted">
    <h3 id="smallheading">
    <p id="para1">
    <p id="para2">
    <p id="para3">[para3] The buttons below demonstrate:(list)</p>
    <ul id="list">
    <div id="nav" class="nav">
  </body>
</html>

```

```

onsuspend null
ontimeout null
ontimeupdate null
onunload null
onvolumechange null
onwaiting null
outerHTML "<p id="para3">[para3] The butt_"
outerText "[para3] The buttons below demo_"
ownerDocument HTMLDocument
pageCount 1
parentElement HTMLBodyElement
parentNode HTMLBodyElement
prefix null
previousElementsibling HTMLParagraphElement
previousSibling Text
properties HTMLPropertiesCollection

```



Creating and Inserting New Elements Using `insertBefore` and `appendChild`

- The second and third rows of the form allow the user to create a new element and insert it before or as a child of the current node, respectively.
- If the user enters text in the second text field and clicks the `Insert Before` button, the text is placed in a new paragraph element, which is inserted into the document before the currently selected element.
- The `Insert Before` button's click event calls function `insert()`.



Creating and Inserting New Elements Using `insertBefore` and `appendChild`

- The `insert()` function calls the `createNewNode()` function, passing it the value of the “ins” input field as an argument.
- The helper function `createNewNode()` creates a paragraph node that contains the text passed to it.

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



Creating and Inserting New Elements Using `insertBefore` and `appendChild`

- Function `createNewNode()` creates a `<p>` element using the document's `createElement` method, which creates a new DOM node, taking the tag name as an argument.
- The `createElement` method creates an element...it does *not* insert the element on the page.

```
// helper function that returns a new paragraph node containing
// a unique id and the given text
function createNewNode( text )
{
    var newNode = document.createElement( "p" );
    nodeId = "new" + idcount;
    ++idcount;
    newNode.setAttribute( "id", nodeId ); // set newNode's id
    text = "[" + nodeId + " ] " + text;
    newNode.appendChild( document.createTextNode( text ) );
    return newNode;
} // end function createNewNode
```



Creating and Inserting New Elements Using `insertBefore` and `appendChild`

- To create the new element, a unique `id` for it is created by concatenating the string “new” with the current value of `idcount`.
- The `setAttribute` function is then called to set the `id` of the new element.
- The value of the text is concatenated with the square brackets used to identify the nodes to the user.
- Then the document’s `createTextNode` method is called to create a node that contains only text. This new node is then used as the argument to the `appendChild` method, which inserts a child node after any existing children of the node on which it is called.



Creating and Inserting New Elements Using `insertBefore` and `appendChild`

- After the `<p>` element is created by `createNewNode` that function returns the new node to the `insert` function, where it's assigned to the variable `newNode`.
- The `newNode` is then inserted before the currently selected node.
- The `parentNode` property contains a node's parent. This property is used in the `insert` function to get the current node's parent. Then the `insertBefore` method is invoked on the parent node with `newNode` and `currentNode` as its arguments. This causes `newNode` to be inserted as a child of the parent directly before `currentNode`.



Creating and Inserting New Elements Using `insertBefore` and `appendChild`

- Finally, the `switchTo` helper function is called to set the highlighted class on the newly created element.
- The input field and button on the third line of the input form allows the user to append a new paragraph node as a child of the current element.
- This is done in a similar manner to the `Insert Before` button's `insert` function. However, in this case the function `appendNode` creates the new node and inserts it as a child of the current node. Examine the JavaScript more closely to see how this mirrors the `insert` function and also how it differs.



[bigheading] HTML5 DOM Tree Demo Page

[smallheading] Element Functionality

[para1] The Document Object Model (DOM) allows for quick, dynamic access to all elements in an HTML5 document for manipulation with JavaScript.

[para2] For more information, check out the "JavaScript lecture notes on the course web site [\[link\]](#) JavaScript - Part 1, 2, and 3.

[para3] The buttons below demonstrate:(list)

- [item1] getElementById and parentNode
- [item2] insertBefore and appendChild
- [item3] replaceChild and removeChild

para3 ←

ically inserted <p> element

Remove Current

Get Parent

Get By id

Insert Before

Append Child

Replace Current

User selects [para3] then enters new text and clicks Insert Before button. HTML effect shown on next page.



Basic DOM Functionality

Temperature Convert... x Dynamic Typing In J... x Way Simple Adding ... x Basic DOM Functiona... x

localhost/Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/JavaScript%20-%20Part%20 JavaScript - Part 1, 2, and 3. Search with Google

[bigheading] HTML5 DOM Tree Demo Page

[smallheading] Element Functionality

[para1] The Document Object Model (DOM) allows for quick, dynamic access to all elements in an HTML5 document for manipulation with JavaScript.

[para2] For more information, check out the "JavaScript lecture notes on the course web site [\[link\]](#) JavaScript - Part 1, 2, and 3.

[new0] This is a new dynamically inserted <p> element

[para3] The buttons below demonstrate:(list)

- [item1] getElementById and parentNode
- [item2] insertBefore and appendChild
- [item3] replaceChild and removeChild

new0	Get By id
ically inserted <p> element	Insert Before
	Append Child
	Replace Current
Remove Current	
Get Parent	



Basic DOM Functionality

Temperature Convert... x Dynamic Typing In J... x Way Simple Adding ... x Basic DOM Functiona... x +

localhost/Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/JavaScript%20-%20Part%202

[bigheading] HTML5 DOM Tree Demo Page

[smallheading] Element Functionality

[para1] The Document Object Model (DOM) allows for quick, dynamic access to all elements in an HTML5 document for manipulation with JavaScript.

[para2] For more information, check out the "JavaScript lecture notes on the course web site [\[link\]](#) JavaScript - Part 1, 2, and 3.

[new0] This is a new dynamically inserted <p> element

[para3] The buttons below demonstrate:(list)

- [item1] getElementById and parentNode

```
<!DOCTYPE html>
<html lang="en">
  <head>
  <body>
    <h1 id="bigheading" class="">
    <h3 id="smallheading">
    <p id="para1">
    <p id="para2">
    <p id="new0" class="highlighted">[new0] This is a new dynamically inserted <p>
    element</p>
    <p id="para3" class="">
    <ul id="list">
    <div id="nav" class="nav">
  </body>
</html>
```

html > body > p#new0.highlighted

childNodes NodeList
children HTMLCollection
classList DOMTokenList
className "highlighted"
clientHeight 19
clientLeft 0
clientTop 0
clientWidth 691
contentEditable "inherit"
currentPage 0
currentStyle CSSStyleDeclaration
dataset DOMStringMap
dir ""
draggable false
dropzone ""
firstChild Text
firstElementChild null
hidden false
id "new0"
innerHTML "[new0] This is a new dynamical"



Replacing and Removing Elements Using `replaceChild` and `removeChild`

- The next two buttons on the input form provide the user with the ability to replace the current element with a new `<p>` element or simply to remove the element entirely.
- When the user clicks the `Replace Current` button, the function `replaceCurrent` is called.
- In function `replaceCurrent`, the `createNewNode` helper function is called in much the same manner as it was when the `InsertBefore` or `AppendChild` buttons were clicked.
- The user's text is retrieved from the input field in the form and the parent of the current node is determined, then the `replaceChild` method is invoked on the parent.



Replacing and Removing Elements Using `replaceChild` and `removeChild`

- The `replaceChild` method takes two arguments, the first of which is the new node to be inserted, and the second is the node to be replaced.

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



[bigheading] HTML5 DOM Tree Demo Page

[smallheading] Element Functionality

[para1] The Document Object Model (DOM) allows for quick, dynamic access to all elements in an HTML5 document for manipulation with JavaScript.

[para2] For more information, check out the "JavaScript lecture notes on the course web site [link] [JavaScript - Part 1, 2, and 3](#).

[para3] The buttons below demonstrate:(list)

- [item1] getElementById and parentNode
- [item2] insertBefore and appendChild
- [item3] replaceChild and removeChild

para1 ← Get By id

Insert Before

Append Child

Replace Current

Remove Current

Get Parent

ically altered by JavaScript!!!

User selects [para1] then enters new text and clicks Replace Current button. HTML effect shown on next page.



[bigheading] HTML5 DOM Tree Demo Page

[smallheading] Element Functionality

[new0] This paragraph has been dynamically altered by JavaScript!!!

[para2] For more information, check out the "JavaScript lecture notes on the course web site [link] [JavaScript - Part 1, 2, and 3](#).

[para3] The buttons below demonstrate:(list)

- [item1] getElementById and parentNode
- [item2] insertBefore and appendChild
- [item3] replaceChild and removeChild

new0	Get By id
	Insert Before
	Append Child
ically altered by JavaScript!!!	Replace Current
Remove Current	
Get Parent	



Basic DOM Functiona... x +

localhost/Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/JavaScript%20-%20Part%20: Search with Google

[bigheading] HTML5 DOM Tree Demo Page

[smallheading] Element Functionality

[new0] This paragraph has been dynamically altered by JavaScript!!!

[para2] For more information, check out the "JavaScript lecture notes on the course web site [link] [JavaScript - Part 1, 2, and 3.](#)

Page 21 The buttons below demonstrate (list)

Documen... Scripts Network Resour... Storage Profiler Errors Utilities Cons...

Basic DOM Functionality

```

<!DOCTYPE html>
<html lang="en">
  <head>
  <body>
    <h1 id="bigheading" class="">
    <h3 id="smallheading">
    <p id="new0" class="highlighted">[new0] This paragraph has been dynamically altered by JavaScript!!!</p>
    <p id="para2">
    <p id="para3">
    <ul id="list">
    <div id="nav" class="nav">
  </body>
</html>

```

Styles Properties Layout Search

accessKey "" align ""

- + all HTMLCollection
- + attributes NamedNodeMap
- + baseURI "file:///localhost/Volumes/RALLY..."
- + childElementCount 0
- + childNodes NodeList
- + children HTMLCollection
- + classList DOMTokenList
- className "highlighted"
- clientHeight 19
- clientLeft 0
- clientTop 0
- clientWidth 558
- contentEditable "inherit"
- currentPage 0
- + currentStyle CSSStyleDeclaration
- + dataset DOMStringMap
- dir ""
- draggable false
- dropzone ""
- + firstChild Text
- firstElementChild null

html > body > p#new0.highlighted



Replacing and Removing Elements Using `replaceChild` and `removeChild`

- Clicking the Remove Current button calls the `remove` function in the JavaScript which removes the currently selected element entirely and highlights the parent.
- If the node's parent is the body element, an error message is displayed to indicate that a top level element cannot be deleted.
- The next page illustrates this error condition.



Basic DOM Functionality

Basic DOM Functiona... x +

Local localhost/Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/JavaScript%20-%20Part%202 Search with Google

[bigheading] Demo Page

<localhost>
Can't remove a top-level element.
 Stop executing scripts on this page
OK

[smallheading] Element Functionality

[para1] The Document Object Model (DOM) allows for quick, dynamic access to all elements in an HTML5 document for manipulation with JavaScript.

[para2] For more information, check out the "JavaScript lecture notes on the course web site [link] JavaScript - Part 1, 2, and 3.

[para3] The buttons below demonstrate:(list)

- [item1] getElementById and parentNode
- [item2] insertBefore and appendChild
- [item3] replaceChild and removeChild

para1 ←

	Get By id
	Insert Before
	Append Child
	Replace Current
Remove Current	
Get Parent	

User selects [para1] then clicks Remove Current button. JavaScript pops up the alert that a top-level element cannot be deleted.



Replacing and Removing Elements Using `replaceChild` and `removeChild`

- In general, `parent.removeChild(child)` looks in a parent's list of children for `child` and removes it.

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



[bigheading] HTML5 DOM Tree Demo Page

[smallheading] Element Functionality

[para1] The Document Object Model (DOM) allows for quick, dynamic access to all elements in an HTML5 document for manipulation with JavaScript.

[para2] For more information, check out the "JavaScript lecture notes on the course web site [link] [JavaScript - Part 1, 2, and 3.](#)

[para3] The buttons below demonstrate:(list)

- [item1] getElementById and parentNode
- [item2] insertBefore and appendChild
- [item3] replaceChild and removeChild

item2 ←

<input type="text"/>	Get By id
<input type="text"/>	Insert Before
<input type="text"/>	Append Child
<input type="text"/>	Replace Current
Remove Current	
Get Parent	

User selects item2] then clicks Remove Current button. HTML effect shown on next page.



Basic DOM Functionality

Basic DOM Functiona... x +

localhost/Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/JavaScript%20-%20Part%201

Search with Google

[bigheading] HTML5 DOM Tree Demo Page

[smallheading] Element Functionality

[para1] The Document Object Model (DOM) allows for quick, dynamic access to all elements in an HTML5 document for manipulation with JavaScript.

[para2] For more information, check out the "JavaScript lecture notes on the course web site [\[link\] JavaScript - Part 1, 2, and 3.](#)

[para3] The buttons below demonstrate:(list)

- [item1] getElementById and parentNode
- [item3] replaceChild and removeChild

list	Get By id
	Insert Before
	Append Child
	Replace Current
Remove Current	
Get Parent	



Determining the Parent Element

- The final piece of functionality in this DOM demo is the button that allows the user to identify the parent of the selected element.
- This is done by calling the `parent` function. This function simply gets the parent node, again making sure its not the body element since we will not allow selecting the entire body element.
- When the parent node is determined, the `switchTo` function is called to highlight the parent node.
- This sequence is illustrated by the next two slides.



[bigheading] HTML5 DOM Tree Demo Page

[smallheading] Element Functionality

[para1] The Document Object Model (DOM) allows for quick, dynamic access to all elements in an HTML5 document for manipulation with JavaScript.

[para2] For more information, check out the "JavaScript lecture notes on the course web site [\[link\]](#) [JavaScript - Part 1, 2, and 3](#).

[para3] The buttons below demonstrate:(list)

- [item1] getElementById and parentNode
- [item2] insertBefore and appendChild
- [item3] replaceChild and removeChild

item2	Get By id
	Insert Before
	Append Child
	Replace Current
Remove Current	
Get Parent	

User selects [itme2] then clicks Get By id. The item2 element is highlighted. Then the user clicks the Get Parent button. HTML effect shown on next page.



[bigheading] HTML5 DOM Tree Demo Page

[smallheading] Element Functionality

[para1] The Document Object Model (DOM) allows for quick, dynamic access to all elements in an HTML5 document for manipulation with JavaScript.

[para2] For more information, check out the "JavaScript lecture notes on the course web site [\[link\]](#) [JavaScript - Part 1, 2, and 3.](#)

[para3] The buttons below demonstrate:(list)

- [item1] getElementById and parentNode
- [item2] insertBefore and appendChild
- [item3] replaceChild and removeChild

list ←

Get By id

Insert Before

Append Child

Replace Current

Remove Current

Get Parent

The parent of [item2] is now highlighted and identified in the get By Id text field.

