CodeAlong:
Modifying the Document Object Model and Adding Events

We'll open a sample HTML page and use the JavaScript console to experiment with parts of the DOM. Notice that any changes to the DOM are live updates and are instantly rendered by the browser in the page.

Sample Page:

Surf to this URL and do a View Source:

http://hucodev.srv.ualberta.ca/hquamen/javascript/dom.html

Here's a version of that page with a display of the DOM elements:

http://hucodev.srv.ualberta.ca/hquamen/javascript/domdisplay.html

Open the JavaScript Console:

Chrome:
- View > Developer > JavaScript Console
- Three-line icon > Tools > JavaScript Console

Safari (note: you may need to activate the Develop menu in Preferences first):
- Develop > Show Error Console

Firefox:
- Tools > Web Developer > Web Console

Internet Explorer (? verify these)
- Press F12
- go through the Tools menu

Changing the DOM Structure with Javascript

In the exercises below, we'll use the console to manipulate the DOM structure and watch the changes take effect in the browser.

First, we'll move a node to a different part of the DOM tree.

```
> p3node = document.getElementById('three')                <-- get a node
> p3node.parentNode.removeChild(p3node)     <-- remove it from the DOM (chaining!)
> s2node = document.getElementById('section2')             <-- find another node
> s2node.appendChild(p3node)                <-- attach the first node to it

> p6node = document.getElementById('six')                  <-- get another node
> p6node.parentNode.removeChild(p6node)                    <-- disconnect it
```

```
> document.getElementById('section1').insertBefore(p6node,
document.getElementById('one'))          <-- insert at the front of section 1; chaining!
```

Create a brand new node, add some attributes, and attach it to the tree:

```
> newpara = document.createElement('p')                    <-- create a new p element
> newpara.id = 'seven'                                     <-- add an ID
> newpara.className = 'container'                          <-- add a class (look at the CSS!)
> newpara.innerText = 'Welcome to the Matrix.'             <-- give it some content
> document.getElementById('section2').appendChild(newpara)    <-- attach it
```

Get arrays of nodes and change all styles in a loop:

```
> anchors = document.getElementsByTagName('a')             <-- get all <a> nodes
> anchors[0]                                               <-- look at zeroth element of array
> for (index in anchors) {                                 <- loop thru and change all classes
      anchors[index].className = 'nounderline';
  }
```

Find a node the harder way: without an ID. Remember that finding by tag name returns an array, but there's only one here, so it must be the zeroth element. Add a CSS style to it. (We've done it here in two steps, but could you do it in one step?)

```
> divnode = document.getElementsByTagName('div')[0]        <-- zeroth in an array
> divnode.className = 'container'                          <-- add a class (look to CSS!)
```

More on NodeLists. They behave like arrays and we can use them exactly as if they were arrays.

```
> children = document.getElementById('section1').childNodes      <-- it's a NodeList
    NodeList(7) [text, p#one, text, p#two, text, p#three, text]   <-- notice text nodes!
> children.length                                          <-- how many children?
> children[1].nodeName                                     <-- It's a P
> children[1].nodeType          <-- It's 1; the Node Type chart says that's an ELEMENT_NODE
```

Look at the attributes of the meta *tag:*

```
> m = document.getElementsByTagName('meta')                <-- it's a NamedNodeMap
> a = m[0].attributes                                      <-- get the zeroth node's attributes in the array
> a[0]                                                     <-- the zeroth attribute
> a[0].name                                                <-- the name of the zeroth attribute
```

```
> a[0].value                <-- the value of the zeroth attribute
> a[0].nodeName             <-- name in the name=value pair
> a[0].nodeType             <-- 2; it's an ATTRIBUTE_NODE
```

Events

Stay on the same page, but you can look at the Event Counter page:
http://hucodev.srv.ualberta.ca/hquamen/javascript/events.html

In the console:

```
> two = document.getElementById('two')          <-- get the DOM element
> two.onclick = function() {alert("You clicked me!");}   <-- attach anonymous function!

> two.onclick                                   <-- this just displays the function
> two.onclick()                                 <-- this calls it

> document.getElementById('one').onmouseover = function() {confirm('Really?');}
```

1. Add to the HTML:

```
<p id=one onclick="makeVowelsBold(this);">This is the first paragraph.</p>
```

2. Next, add a handler function to an external **.js** file:

```
function makeVowelsBold(elem) {
        elem.innerHTML = elem.innerHTML.replace(/([aeiou])/ig, '<b>$1</b>');
}
```

3. Make it toggle:

```
var bold = false;

function makeVowelsBold(elem) {
      if (!bold) {
            // bold is false, so make bold
```

```
            elem.innerHTML = elem.innerHTML.replace(/([aeiou])/ig, '<b>$1</b>');
            bold = true;
        } else {
            // bold is true, so unbold
            elem.innerHTML = elem.innerHTML.replace(/<b>([aeiou])<\/b>/ig, '$1');
            bold = false;
        }
}
```