Introduction to Programming

Andrew Butterfield

Department of Computer Science
School of Computer Science and Statistics
Trinity College, Dublin

January 14th, 2008
In Javascript the HTML page the script is running in is exposed to the programmer through a standard mechanism called the “Document Object Model” (or “DOM”).

We have already encountered the DOM in a brief way.

document.write("Hello");

Is the invocation of a method in the “document” object, which models the HTML document that the script is associated with. Older browsers offered various, incompatible, ways to access elements of the document, which made it difficult to write Javascript programs which would work reliably. The DOM is an attempt to standardize the methods and attributes of the document model.
This HTML document:

```html
<body id="bodyNode">
  <p>Hello world</p>
  doc.body
  <p></p>
</body>
```

Will produce a *document tree* similar to this:
We can use methods and attributes of the document object to obtain objects that correspond to elements of the document. What is far more interesting is that modifying those objects will cause corresponding changes in the browser window.

```javascript
n = document.getElementById("bodyNode").childNodes[1];
n.nodeValue = "blah blah";
n.style.fontFamily = "Arial";
```
Each node contains an array: `childNodes`, which contains the immediate children.
As an alternative, there are also attributes —`firstChild`— (and `lastChild`) which can be used to find the first (and last) sub-nodes, and each node also has `nextSibling` and `previousSibling` attributes which can be used to cycle through each child in turn.

This gives multiple ways of traversing the document; we can choose the appropriate one for the program we are writing.
Example: counting nodes in the document

We can write a *recursive* function to traverse the document tree:

A recursive function is one which fulfills its goal by subdividing the problem into a simple step and a sub-task which can be performed by a “nested” invocation of the original function. This is a natural pattern for working with the nested structure of the document tree.

The function will contain an invocation of *itself* - but the self-invocation will work on a different (usually smaller) set of data.
function walk(n) {
    var tmp = n.firstChild;
    var result = 1;
    while( tmp ){
        result += walk(tmp);
        tmp = tmp.nextSibling;
    }
    return result;
}

document.write( "Total number of nodes: "+walk( document ) );