The Learning House

427 S. 4th Street #300

Louisville KY 40202

Document Object Model

.NET Cohort

Coding Bootcamp

SOFTWARE GUILD

What is the DOM?

The Document Object Model (DOM) is an API for HTML and XML documents.

The DOM exposes a document as a tree of nodes, which allows you as a developer to add, remove, and modify individual nodes on a page.

Recent versions of browsers all have good DOM implementations, older versions do not, and this can cause headaches.

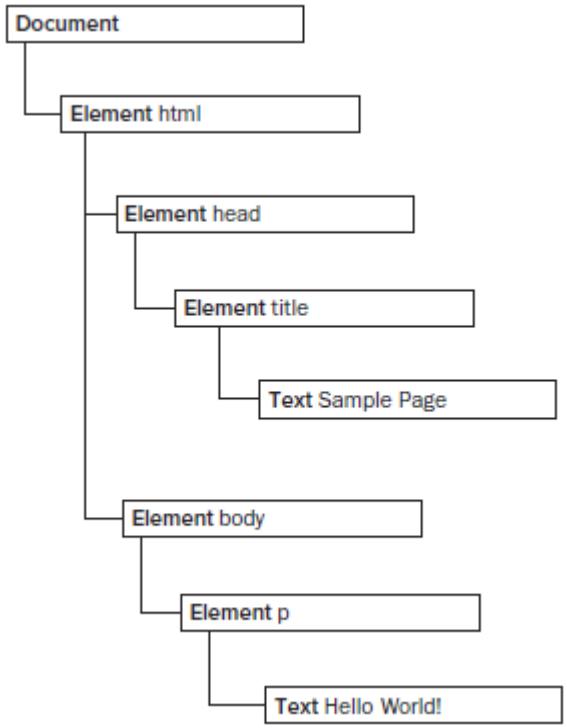
A Hierarchy of Nodes

When we say a hierarchy of nodes, it means that each HTML (or XML) tag has different characteristics and can often contain other tags.

The relationships between tags create a tree, rooted at a particular node.

Consider the HTML below and how it breaks down into parent and child nodes.

```
<html>
    <head>
        <title>Sample Page</title>
    </head>
    <body>
        <p>Hello World!</p>
    </body>
</html>
```



The Document Type

JavaScript represents document nodes via the document type. The document object is an instance of HTMLDocument and represents the entire page.

document is a property of window so it is accessible globally.

We can use the document to get references to elements or sets of elements using the getElementById() and getElementsByTagName() methods

```
//retrieve reference to the <div> with id "headerDiv"
var div = document.getElementById("headerDiv");

// retrieve all images in the page
var images = document.getElementsByTagName("img");

//output the number of images
alert(images.length);

//output the src attribute of the first image
alert(images[0].src);

//output the src attribute of the first image
alert(images.item(0).src);

// select image with specific name
var myImage = images["imgName"];
```

Gotchas

IE 7 and earlier the id was case-insensitive

Additionally IE7 also return form elements (input, button, select, etc) whose name matches the id.

So make sure your form fields don’t have name attributes equal to any ids (unless it’s the same element)

getElementsByName()

document.getElementsByName() will select all the elements with a specified name.

This is generally only used with radio buttons, since they have to share the same name for the model selection to work properly.

```
<fieldset>
    <legend>Which color do you prefer?</legend>
    <ul>
        <li>
            <input type="radio" value="red" name="color" id="colorRed">
            <label for="colorRed">Red</label>
        </li>
        <li>
            <input type="radio" value="green" name="color" id="colorGreen">
            <label for="colorGreen">Green</label>
        </li>
        <li>
            <input type="radio" value="blue" name="color" id="colorBlue">
            <label for="colorBlue">Blue</label>
        </li>
    </ul>
</fieldset>

<script>
    //retrieve reference to all the radio buttons
    var radios = document.getElementsByName("color");
</script>
```

document.write

If you embed a script tag in the middle of the markup you can use the document.write() method to write out dynamic markup.

This typically isn’t used much in MVC because we have razor, which is better.

Manipulating HTML Attributes of a selected element

The HTMLElement object returned by a document lookup has some useful members that allow you to retrieve and set common attributes of that object.

We can also use:

- getAttribute()
- setAttribute()
- removeAttribute()

For these to pass in the string attribute you are looking for. To set an attribute, pass the attribute name as the first param and the value as the second.

Ex: div.setAttribute("class", "ft");

Note that the style attribute returns an object in some instances!

```
<div id="myDiv" class="bd" title="Body text" lang="en" dir="ltr"></div>

<script>
    var div = document.getElementById("myDiv");
    alert(div.id); //"myDiv"
    alert(div.className); //"bd"
    alert(div.title); //"Body text"
    alert(div.lang); //"en"
    alert(div.dir); //"ltr"

    div.id = "someOtherId";
    div.className = "ft";
    div.title = "Some other text";
    div.lang = "fr";
    div.dir ="rtl";

</script>
```

Creating and placing elements in the DOM

New elements can be created by the document.createElement() method. The method accepts the tag name of the element to create.

If you want to just place text, you can use the document.createTextNode() method.

To place an element into the hierarchy you can use the appendChild() method.

```
<script>
    var element = document.createElement("div");
    element.className = "message";

    var textNode = document.createTextNode("Hello world!");
    element.appendChild(textNode);

    document.body.appendChild(element);
</script>
```

querySelector() and querySelectorAll()

The last two methods we want to demonstrate are the querySelector() and querySelectorAll() methods.

Both of these methods take CSS queries as parameters and return either the first match, or all of the matches as a node list.

```
<script>
    //get the body element
    var body = document.querySelector("body");

    //get the element with the ID "myDiv"
    var myDiv = document.querySelector("#myDiv");

    //get first element with a class of "selected"
    var selected = document.querySelector(".selected");

    //get first image with class of "button"
    var img = document.body.querySelector("img.button");

    //get all elements with class of "selected"
    var selecteds = document.querySelectorAll(".selected");

    //get all <strong> elements inside of <p> elements
    var strongs = document.querySelectorAll("p strong");
</script>
```

Summary

There are a lot more methods and techniques for dealing with the DOM. That all being said, most web developers use jQuery for DOM manipulation… just be aware that under the covers of jQuery is a lot of plain old JavaScript.

If you're just doing something quick and simple, the overhead of jQuery should be avoided.