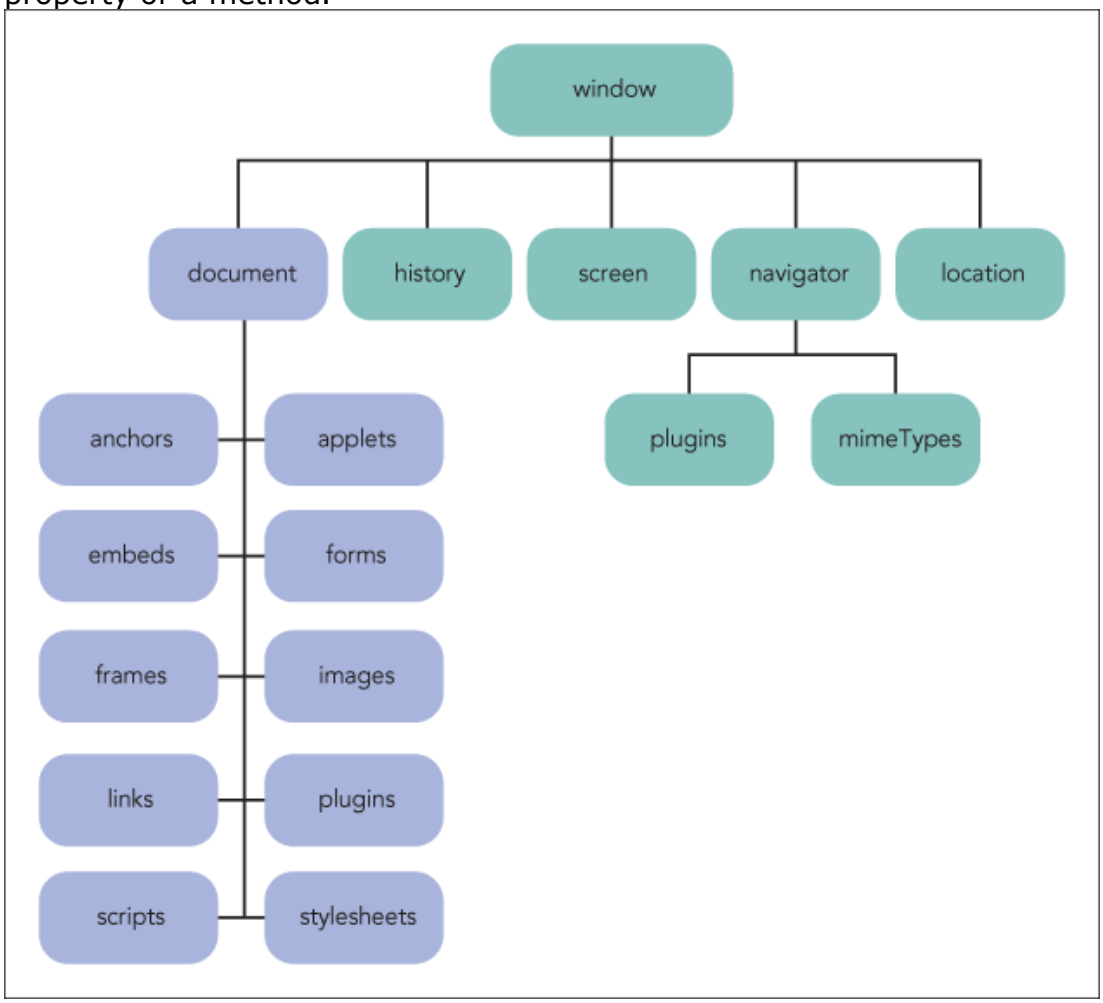
The DOM - Naming Objects in the DOM

The BOM stands for the **Browser Object Model**
The DOM stands for the **Document Object Model** (and a subtree of the BOM)

The term **browser** is the browser window, and **document** is the web page in the browser window. The **window** of the object tree below is the browser window. It is the root of the tree and the tree is the **Brower Object Model (BOM)**. The `window` and the `document` are objects as are the other nodes[1] appearing in the object tree below. They are all objects in the BOM tree. Recall an object is a data structure that has properties and has methods that can act on the object. It is expressed with a dotted notation. For example, there is `document.write()` and `myDate.getMonth()`. write() and `getMonth()` are methods. Here is an array object with a property called length: `myArr.length`. The expression consists of the object, followed by a dot, followed by either a property or a method.



Now, with the BOM, we name an object in it by starting by naming the window object at the top and then traversing downwards, over the branches. As we traverse, we name each object in sequence long a path. The branch between 2 objects will correspond to where a dot would be placed in the named object expression. More than one dot is allowed.

[1] A node is an item in the tree and the branches connect the items.

Study very carefully the opening pages of the session 9.2 in the textbook, specifically pages 684-691 (session __ in the MindTap Reader). It explains how the browser window is an object and how it contains other objects, including the document object (DOM) that branches off the window. And the document likewise contains objects that are one in the same as the many elements in can exist in a web page. For example, the form (named `forms` in the tree) is an object and the form elements it contains are also objects. Although not listed as a node in this version of the tree, the forms' elements (checkboxes, textboxes, etc) collectively are called the `elements` object and would appear as a node off the forms object node. And the document elements are themselves objects. Observe that they are in the tree and branching off the `document` object. Some of them include `anchors`, `images`, and `links`. `Anchors` correspond to the `<a>` element and `links` to the `<link>` element in the web page. Objects are named with the following form:

`object1.object2.object3. …`

where `object2` is a child of `object1` and `object3` is a child of `object2` and so on. The dot corresponds to a branch, and the naming is from top, going down the tree, branch by branch.

Recall that a child node of the tree branches directly off its parent node.

For example, the following names a collection of image objects in the document, defined as an *object collection*:

```
window.document.images
```

You can drop off the word 'window' in any of these expressions (since it's assumed). Here, you can simply name it as:

```
document.images
```

So, this references all the image objects in the page (i.e. all the <img> tags in the page). This will allow for the coder to manipulate this image element in the page with JavaScript script.

The next expression names a specific image in this collection with `id logo1` (or `name logo1`):

```
document.images.logo1
```

And this names the image's source file, that is its `src` attribute:

```
document.images.logo1.src
```

But if we are looking at image as an object now, we call it a "property" instead of an "attribute". So many attributes of elements are now properties of the elements now considered as objects.

And furthermore, we can assign values to this object variable. (Yes, this named expression is a variable.)

So now we see that the page elements are objects of JavaScript. And related to that, it is also the case that the page elements' attributes are properties of the objects. Looking at this idea, let's assign a new image file called "`myimage.jpg`" as its source file:

```
document.images.logo1.src = "myimage.jpg";
```

One more way we can name the same image object is as:

`document.getElementById("logo1")`

`getElementById()` is a method of the `document` object. This expression names the element in the page that have `id="logo1"`.

Here is another similar method:

`getElementsByTagName(tagname)` where `tagname` is the element's tagname.

For example: `document.getElementsByTagName("h1")` will return an object collection of h1 objects.

Exercise:
Look at the code called **prac_DOM.html.** There is an event handler in this code. Load the web page and try mousing over the picture in the page and see what happens. Then try mousing off the picture. See if you can follow the code that performs this action. But first take a look at a brief explanation about events next.

An event is an action by a user visiting the webpag e. For example, he clicks a button in it. An event handler will be an attribute of this button element (and JavaScript object). Some examples of event handlers are `onclick` and `onmouseover`.

The event handler is assigned script as its value. For the `onclick` handler, when the user clicks on the element containing it, the script it is assigned will execute.

For example, `<a onlclick="MyFunction();">` causes the following to occur. When the user clicks on this link, the function called `MyFunction();` will run.

Optional: And yet other ways to name objects in the web page.

There are at least 2 other ways to name objects in the `document`. There is one that uses array notation, and another that uses the element's `name` attribute.

Naming using arrays: The JavaScript language inherently creates arrays of each of the various object collection in the tree. So, the `forms` object collection name is also the name of a JavaScript array. The array contains the object collection. Just as a variable name is used to store a single value, an array name is used to store more than one value. These arrays can be named and allow another way

to name expressions. The array `forms` contains all the forms in the current document, where the 1st form in a web page `document` is named `forms[0]`, the 2nd form in the `document` is `forms[1]`, the 3rd is `forms[2]` and so on. Likewise, there are arrays for the `images` object collection, and other collections.

Naming using the `name` attribute: An example of naming using the `name` attribute follows:

Suppose you have a form called `myForm`:

```
<form name="myForm" action="" method="">
     <input type="text" name="address"  value="myAddress"/>
     <input type="text" name="city" id="myCity" />
<input type="submit" value="submit" />
</form>
```

Then `input` element with `name="address"` can be named as:

```
window.document.myForm.address
```