# JavaScript Document Object Model. 12th Day (12%)

A Document object represents the HTML document that is displayed in that window. The Document object has various properties that refer to other objects which allow access to and modification of document content.
The way a document content is accessed and modified is called the **Document Object Model**, or **DOM**. The Objects are organized in a hierarchy. This hierarchical structure applies to the organization of objects in a Web document.

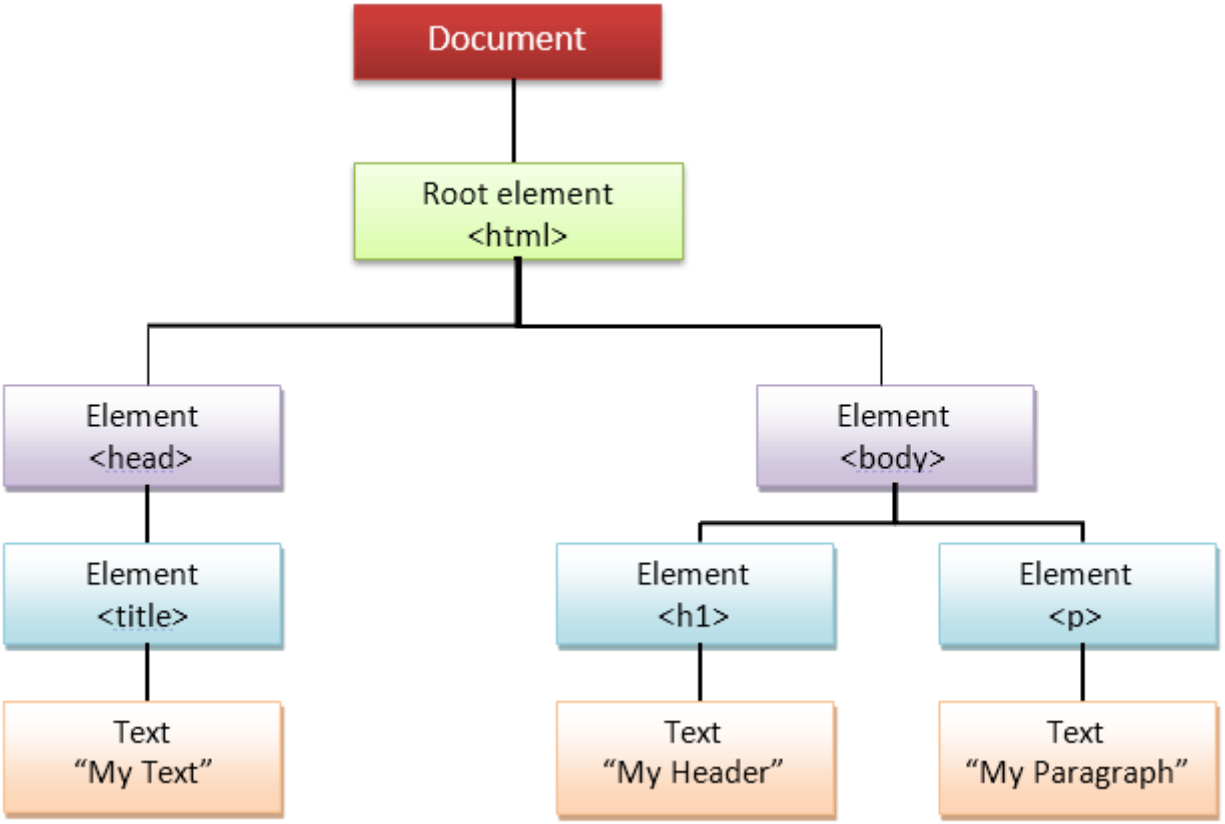
**Window object** – Top of the hierarchy. It is the outmost element of the object hierarchy.

**Document object** – Each HTML document that gets loaded into a window becomes a document object. The document contains the contents of the page.

**Form object** – Everything enclosed in the <form>…</form> tags sets the form object.

**Form control elements** – The form object contains all the elements defined for that object such as text fields, buttons, radio buttons, and checkboxes.

JavaScript can access all the elements in a webpage making use of Document Object Model (DOM). In fact, the web browser creates a DOM of the webpage when the page is loaded. The DOM model is created as a tree of objects like this:




In General, the Document Object Modelis a way to manipulate the structure and style of an HTML page. It represents the internals of the page as the browser sees it, and allows the developer to alter it with JavaScript.

**Trees and Branches**

HTML is an XML–like structure in that the elements form a structure of parents’ nodes with children, like the branches of a tree. There is one root element（html）with branches like head and body, each with their own branches. For this reason, the DOM is also called the **DOM tree**.

Modifying the DOM, by picking an element and changing something about it, is something done often in JavaScript. To access the DOM from JavaScript, the document object is used. It’s provided by the browser and allows code on the page to interact with the content.

**Getting an Element from the DOM**.

The first thing to know is how to get an element. There are a number of ways of doing it, and browsers support different ones. Starting with the best supported we'll move through to the latest, and most useful, versions.

**- By ID**

document.getElementById is a method for getting hold of an element – unsurprisingly – by its ID.

```
var pageHeader = document.getElementById('page-header');
```

The pageHeader element can then be manipulated – its size and color can be changed, and other code can be declared to handle the element being clicked on or hovered over. It's supported in pretty much all the browsers you need to worry about.

**By Tag Name**

document.getElementsByTagName works in much the same way as getElementById, except that it takes a tag name (a, ul, li, etc) instead of an ID and returns a **NodeList**, which is essentially an array of the DOM Elements.

**By Class Name**

document.getElementsByClassName returns the same kind of NodeList as getElementsByTagName, except that you pass a class name to be matched, not a tag name.

**By CSS Selector**

A couple of new methods are available in modern browsers that make selecting elements easier by allowing the use of CSS selectors. They are document.querySelector and document.querySelectorAll.

```
var pageHeader = document.querySelector('#header');
var buttons = document.querySelectorAll(.btn);
```

querySelector, like getElementById, returns only one element whereas querySelectorAll returns a NodeList. If multiple elements match the selector you pass to querySelector, only the first will be returned.

If you'd like to have a look at the DOM for a page, open up the developer tools in your browser and look for the "elements" pane. It's a great insight into how the browser thinks, and in most browsers you can remove and modify elements directly. Give it a try!

**DOM compatibility**.

If you want to write a script with the flexibility to use either W3C DOM or IE 4 DOM depending on their availability, then you can use a capability–testing approach that first checks for the existence of a method or property to determine whether the browser has the capability you desire. For example –

```
if (document.getElementById) {
   // If the W3C method exists, use it
} else if (document.all) {
   // If the all[] array exists, use it
} else {
   // Otherwise use the legacy DOM
}
```

**Best Wishes**.

**Lux Tech Academy**.