**The DOM**

When you open any webpage in a browser, the HTML of the page is loaded and rendered visually on the screen.  
To accomplish that, the browser builds the **Document Object Model** of that page, which is an object oriented model of its logical structure.  
The DOM of an HTML document can be represented as a nested set of boxes:

**DOM Tree**

The DOM represents a document as a tree structure.  
HTML elements become interrelated **nodes** in the tree.  
All those nodes in the tree have some kind of relations among each other.   
Nodes can have **child** nodes. Nodes on the same tree level are called **siblings**.  
For example, consider the following structure:

**The document Object**

There is a predefined **document** object in JavaScript, which can be used to access all elements on the DOM.  
In other words, the **document** object is the owner (or **root**) of all objects in your webpage.  
So, if you want to access objects in an HTML page, you always start with accessing the document object.  
**For example:** document.body.**innerHTML** = "Some text";  
As **body** is an element of the DOM, we can access it using the **document** object and change the content of the **innerHTML** property.  
The **innerHTML** property can be used on almost all HTML elements to change its content

**Selecting Elements**

All HTML elements are **objects**. And as we know every object has **properties** and **methods**.  
The **document** object has methods that allow you to select the desired HTML element.  
These three methods are the most commonly used for selecting HTML elements: //finds element by id  
document.**getElementById**(id)   
  
//finds elements by class name  
document.**getElementsByClassName**(name)   
  
//finds elements by tag name  
document.**getElementsByTagName**(name)

In the example below, the **getElementById** method is used to select the element with **id="demo"** and change its content: var elem = document.**getElementById**("demo");  
elem.innerHTML = "Hello World!";

**Selecting Elements**

The **getElementsByClassName**() method returns a collection of all elements in the document with the specified class name.  
For example, if our HTML page contained three elements with class="demo", the following code would return all those elements as an array: var arr = document.**getElementsByClassName**("demo");  
//accessing the second element  
arr[1].innerHTML = "Hi";  
Similarly, the **getElementsByTagName** method returns all of the elements of the specified tag name as an array.  
The following example gets all paragraph elements of the page and changes their content: <p>hi</p>  
<p>hello</p>  
<p>hi</p>  
<script>  
var arr = document.getElementsByTagName("p");  
for (var x = 0; x < arr.length; x++) {  
arr[x].innerHTML = "Hi there";  
}  
</script>  
The script will result in the following HTML:<p>Hi there</p>  
<p>Hi there</p>  
<p>Hi there</p>  
We used the **length** property of the array to loop through all the selected elements in the above example.

**Working with DOM**

Each element in the DOM has a set of properties and methods that provide information about their relationships in the DOM:  
element.**childNodes** returns an array of an element's child nodes.  
element.**firstChild** returns the first child node of an element.  
element.**lastChild** returns the last child node of an element.  
element.**hasChildNodes** returns true if an element has any child nodes, otherwise false.  
element.**nextSibling** returns the next node at the same tree level.  
element.**previousSibling** returns the previous node at the same tree level.  
element.**parentNode** returns the parent node of an element.

We can, for example, select all child nodes of an element and change their content:

<html>  
<body>  
<div id ="demo">  
<p>some text</p>  
<p>some other text</p>  
</div>  
  
<script>  
var a = document.getElementById("demo");  
var arr = a.childNodes;  
for(var x=0;x<arr.length;x++) {  
arr[x].innerHTML = "new text";  
}  
</script>  
  
</body>  
</html>

**Changing Attributes**

Once you have selected the element(s) you want to work with, you can change their attributes.   
As we have seen in the previous lessons, we can change the text content of an element using the **innerHTML** property.  
Similarly, we can change the attributes of elements.  
For example, we can change the **src** attribute of an image: <img id="myimg" src="orange.png" alt="" />  
<script>  
var el = document.getElementById("myimg");  
el.**src** = "apple.png";  
</script>  
We can change the **href** attribute of a link:

<a href="http://www.example.com">Some link</a>  
<script>  
var el = document.getElementsByTagName("a");  
el[0].**href** = "http://www.sololearn.com";  
</script>

**Changing Style**

The style of HTML elements can also be changed using JavaScript.  
All style attributes can be accessed using the **style** object of the element.   
**For example:**

<div id="demo" style="width:200px">some text</div>  
<script>  
var x = document.getElementById("demo");  
**x.style.color** = "6600FF";  
**x.style.width** = "100px";  
</script>

An alternative way of achieving the same result would be the use of the **parentNode** property to get the parent of the element we want to remove:  
var child = document.getElementById("p1");  
child.**parentNode**.removeChild(child);

**Removing Elements**

To remove an HTML element, you must select the parent of the element and use the **removeChild**(node) method.  
**For example:**

<div id="demo">  
<p id="p1">This is a paragraph.</p>  
<p id="p2">This is another paragraph.</p>  
</div>  
  
<script>  
var parent = document.getElementById("demo");  
var child = document.getElementById("p1");  
parent.**removeChild**(child);  
</script>

**Replacing Elements**

To replace an HTML element, the element.**replaceChild**(newNode, oldNode) method is used.  
**For example:**

<div id="demo">  
<p id="p1">This is a paragraph.</p>  
<p id="p2">This is another paragraph.</p>  
</div>  
  
<script>  
var p = document.createElement("p");  
var node = document.createTextNode("This is new");  
p.appendChild(node);  
  
var parent = document.getElementById("demo");  
var child = document.getElementById("p1");  
parent.**replaceChild**(p, child);  
</script>