# DOM Manipulation

Using JavaScript you can create dynamic HTML (A **dynamic** programming language is one in which operations are otherwise done at compile-time can be also done at run-time.)

**D**ocument **O**bject **M**odel of HTML page (constructs a tree of objects…[see w3schools example](https://www.w3schools.com/js/js_htmldom.asp)).

Think about every piece on the tree as an *object with properties and methods*. Using dot notation, you can access properties of or apply methods to objects.

The object is **document**, so:

* document**.getElementById** is the object.**METHOD**
* When you add **.innerHTML** (document.getElementById.innerHTML = ….) you are accessing an element and changing a **PROPERTY** of that element.

With JavaScript you can make these adjustments to the DOM:

**Change HTML elements (<body>, <title>, <a>, <p>, <h1>)**

First find the element(s), then change it(them).

Example 1:

var parent1 = document.getElementById("div1");

var child1 = document.getElementById("p1");

parent1.removeChild(child1);

(used dot notation to access an object(parent1), then apply a method to it(removeChild). We removed the child1 object from the parent1 object.)

Methods to FIND elements: usually these elements have been identified as an object(element1) with a unique id(“div1”) so *var parent1 = document.getElementById("div1");*

* document.getElementByID(“id”)-- document.getElementById(“div1”) would find <div id=“div1”>
* document.getElementByTagName(“name”)-- document.getElementByTagName(“p”)[0] would find all <p> then select the first [0] <p> ---can add a specific index number to get a specific tag element. Could be “div”, “body”, “img”, “a”, etc…
* document.getElementByClassName(“name”) used mostly in CSS-- document.getElementByClassName(“intro”) would find all tags that have <*element* class=”intro”>

Methods to add or remove elements. You can add/remove elements to the document, or you can be more specific and add/remove elements to *a parent object inside the document*… *parent1.removeChild(child2); see the full example in example 1 above*

* document.createElement(element) document.createElement(“p”) creates a <p> element and adds it to the HTML document
* document.removeChild(element) document.removeChild(“p”) removes a <p> element, you would need more specific identifiers here unless you only have one <p> element in the doc
* document.appendChild(element) would add the specified element to the end of the HTML doc
* document.replaceChild(new, old)
* document.write(text)

**Change HTML attributes** by accessing HTML DOM **PROPERTIES** (href, src, width and height). Using JS we can access and change the values of an HTML element. In every case, substitute “element” with a specific element such as <p>, <img>, <a>…

* element.innerHTML = “*new HTML content*”—h1.innerHTML = “This is a newTitle!”;
* element.attribute = new value--img.src = “newPicture.jpg”;
* element.style.property = new style—button.style.color = 'white';
* (method) element.setAttribute(attribute, value)—similar to element.attribute above. Img.setAttribute(src, “newPicture.jpg”);

**Change CSS styles** (like color, fontFamily, FontSize)

document.getElementById('myBtn').style.color = 'white';

**Create or React to HTML Events** (for a review of these see [w3schools](https://www.w3schools.com/tags/ref_eventattributes.asp))

document.getElementById(id).onclick = function() { code}

document.getElementById("myBtn2").addEventListener("click", function addIt() {

var p1 = document.createElement("p");

var n1 = document.createTextNode("Correct answer: 1, 3, 5, 19, 67, 89.");

p1.appendChild(n1);

var element = document.getElementById("div1");

element.appendChild(p1);

});