

www.agilityio.com

JavaScript Notes

**Docs: DOM (Document Object Model)**

[**Chapter 1: Introduction to the DOM**](https://www.digitalocean.com/community/tutorials/introduction-to-the-dom)

**What is the DOM?**

* A website consists of an HTML document. After parsing the style and structure of the HTML and CSS, the browser creates a representation of the document known as the Document Object Model. This model allows JavaScript to access the text content and elements of the website document as objects.

**The Document Object**

* A document object is a built-in object with many properties and methods we can use to access and modify websites.
* In Developer Tools on index.html, move to the Console tab. Type the document into the console and press ENTER. You will see that what is output is the same as what you see in the Elements tab.

**What is the Difference Between the DOM and HTML Source Code?**

* There are two instances in which the browser-generated DOM will be different from the HTML source code:
* The DOM is modified by client-side JavaScript.
* The browser automatically fixes errors in the source code.

[**Chapter 2: How To Modify Attributes, Classes, and Styles in the DOM**](https://www.digitalocean.com/community/tutorials/how-to-modify-attributes-classes-and-styles-in-the-dom)

**Review of Selecting Elements**

* Until recently, a popular JavaScript library called jQuery was most often used to select and modify elements in the DOM. jQuery simplifies selecting one or more elements and applying changes to all of them simultaneously. To access a single element, use the methods document.querySelector() and document.getElementById(). The querySelector () method is more robust because it can select an element on the page with any type-of selector.
* However, when accessing multiple elements by a common selector, such as a specific class, you have to loop through all the elements in the list.

**Note:**

The methods getElementsByClassName() and getElementsByTagName() will return HTML collections which do not have access to the forEach() method that querySelectorAll() has. In these cases, you will need to use a standard for loop to iterate through the collection.

* This is one of the most important differences to be aware of when progressing from jQuery to vanilla JavaScript. It is important to note the process of applying those methods and properties to multiple elements.

**Modifying Attributes**

* Attributes are values that contain additional information about HTML elements. They usually come in name/value pairs and may be essential depending on the element.
* Some of the most common HTML attributes are the src attribute of an IMG tag, the Href of an A tag, and the class, id, and style attributes. View the attribute list on the Mozilla Developer Network for a full list of HTML attributes. Custom elements not part of the HTML standard will be prepended with data- or aria-.
* Any attribute can be edited this way as well as with the above methods.
* The hasAttribute() and getAttribute() methods are usually used with conditional statements, and the setAttribute() and removeAttribute() methods are used to directly modify the DOM.

**Modifying Classes**

* The class attribute corresponds to CSS class selectors.
* CSS classes are used to apply styles to multiple elements, unlike IDs which can only exist once per page. In JavaScript, we have the className and classList properties to work with the class attribute.
* **Note:**  If any classes already exist on the element, this will override them. You can add multiple space-delimited classes using the className property, or use it without assignment operators to get the current value of the class on the element.
* The other way to modify classes is via the classList property, which comes with a few helpful methods. These methods are similar to the jQuery addClass, removeClass, and toggleClass methods.
* Unlike in the className example, using classList.add() will add a new class to the list of existing classes. You can also add multiple classes as comma-separated strings. It is also possible to use setAttribute to modify the class of an element.

**Modifying Styles**

* The style property represents the inline styles on an HTML element.
* One option to edit the styles is with setAttribute().
* However, this will remove all existing inline styles from the element. Since this is likely not the intended effect, it is better to use the style attribute directly
* CSS properties are written in kebab-case, which is lowercase words separated by dashes. However, kebab-case CSS properties cannot be used on the JavaScript style property, as the dash - is used for subtraction. Instead, they will be replaced with their camelCase equivalent, which is when the first word is lowercase, and all subsequent words are capitalized. In other words, instead of text-align use textAlign for the JavaScript style property.