**Dom Node :**

* According to the W3C HTML DOM standard, everything in an HTML document is a node:
* The entire document is a document node
* Every HTML element is an element node
* The text inside HTML elements are text nodes
* Every HTML attribute is an attribute node (deprecated)
* All comments are comment nodes
* With the HTML DOM, all nodes in the node tree can be accessed by JavaScript.
* New nodes can be created, and all nodes can be modified or deleted.
* Creating New HTML Elements (Nodes)
* To add a new element to the HTML DOM, you must create the element (element node) first, and then append it to an existing element.
* Removing Existing HTML Elements
* To remove an HTML element, use the remove() method:
* const elmnt = document.getElementById("p1"); elmnt.remove();
* **NOTE :** For browsers that does not support the remove() method, you have to find the parent node to remove
* const parent = document.getElementById("div1");  
  const child = document.getElementById("p1");
* parent.removeChild(child);

**Node relation :**

* The nodes in the node tree have a hierarchical relationship to each other.
* The terms parent, child, and sibling are used to describe the relationships.
* In a node tree, the top node is called the root (or root node)
* Every node has exactly one parent, except the root (which has no parent)
* A node can have a number of children
* Siblings (brothers or sisters) are nodes with the same parent

**Node Name :**

* The nodeName property specifies the name of a node.
* nodeName is read-only
* nodeName of an element node is the same as the tag name
* nodeName of an attribute node is the attribute name
* nodeName of a text node is always #text
* nodeName of the document node is always #document
* Note: nodeName always contains the uppercase tag name of an HTML element**.**

**Node List :**

* A NodeList object is a list (collection) of nodes extracted from a document.
* A NodeList object is almost the same as an HTMLCollection object.
* Some (older) browsers return a NodeList object instead of an HTMLCollection for methods like getElementsByClassName().
* All browsers return a NodeList object for the property childNodes.
* Most browsers return a NodeList object for the method querySelectorAll().
* **Not an Array!**
* A NodeList may look like an array, but it is not.
* You can loop through a NodeList and refer to its nodes by index.
* But, you cannot use Array methods like push(), pop(), or join() on a NodeList.

**Dom collection :**

* The getElementsByTagName() method returns an HTMLCollection object.
* An HTMLCollection object is an array-like list (collection) of HTML elements.
* **An HTMLCollection is NOT an array!**
* An HTMLCollection may look like an array, but it is not.
* You can loop through the list and refer to the elements with a number (just like an array).
* However, you cannot use array methods like valueOf(), pop(), push(), or join() on an HTMLCollection.
* indexوسيلة اختيار عنصر بطريقه ال
* const myCollection = document.getElementsByTagName("p");  
  for (let i = 0; i < myCollection.length; i++) {  
    myCollection[i].style.color = "red";  
  }

**Difference**

* A **NodeList** and an **HTMLcollection** is very much the same thing.
* Both are array-like collections (lists) of nodes (elements) extracted from a document. The nodes can be accessed by index numbers. The index starts at 0.
* Both have a **length** property that returns the number of elements in the list (collection).
* An HTMLCollection is a collection of **document elements**.
* A NodeList is a collection of **document nodes** (element nodes, attribute nodes, and text nodes).
* HTMLCollection items can be accessed by their name, id, or index number.
* NodeList items can only be accessed by their index number.
* An HTMLCollection is always a **live** collection. Example: If you add a <li> element to a list in the DOM, the list in the HTMLCollection will also change.
* A NodeList is most often a **static** collection. Example: If you add a <li> element to a list in the DOM, the list in NodeList will not change.
* The getElementsByClassName() and getElementsByTagName() methods return a live HTMLCollection.
* The querySelectorAll() method returns a static NodeList.
* The childNodes property returns a live NodeList.

## Overview of Append()

The append() method is a relatively recent addition to JavaScript and provides a convenient way to add elements or strings to the end of a parent element. It accepts one or more arguments, which can be DOM elements, text strings, or a mix of both. The append() method appends the provided content as the last child of the parent element.

## Overview of appendChild()

The appendChild() method has been available in JavaScript for a long time and is used to add a single child node to a parent element. It accepts a single argument, which is the node to be added as the last child of the parent element. The argument must be a valid DOM element or a text node.

Differences between append() and appendChild()

Although both append() and appendChild() serve the purpose of adding elements to a parent element, there are some important differences to note:

* **Number of Arguments:** append() can accept multiple arguments, allowing you to add multiple elements or strings at once. On the other hand, appendChild() can only accept a single argument, which represents the node to be added.
* **Type Flexibility:** append() can handle a mix of DOM elements and text strings as arguments. It automatically converts text strings into text nodes before appending them to the parent element. Conversely, appendChild() can only accept DOM elements or text nodes.
* **Chaining Support:** append() returns the modified parent element itself, which allows you to chain additional method calls. For example, you can append multiple elements in a single statement. appendChild(), however, does not return anything, making it unsuitable for chaining method calls.