[Difference between Node object and Element object?](https://stackoverflow.com/questions/9979172/difference-between-node-object-and-element-object)

A node is the generic name for any type of object in the DOM hierarchy. A node could be one of the built-in DOM elements such as document or document.body, it could be an HTML tag specified in the HTML such as <input> or <p> or it could be a text node that is created by the system to hold a block of text inside another element. So, in a nutshell, a node is any DOM object.

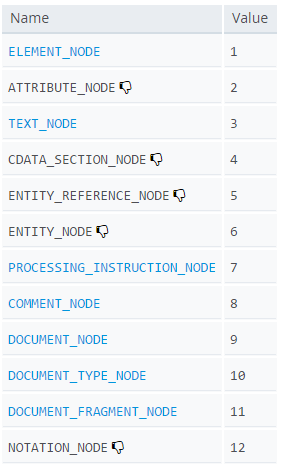
An element is one specific type of node as there are many other types of nodes (text nodes, comment nodes, document nodes, etc...).

The DOM consists of a hierarchy of nodes where each node can have a parent, a list of child nodes and a nextSibling and previousSibling. That structure forms a tree-like hierarchy. The documentnode would have its list of child nodes (the head node and the body node). The body node would have its list of child nodes (the top level elements in your HTML page) and so on.

So, a nodeList is simply an array-like list of nodes.

An element is a specific type of node, one that can be directly specified in the HTML with an HTML tag and can have properties like an id or a class. can have children, etc... There are other types of nodes such as comment nodes, text nodes, etc... with different characteristics. Each node has a property .nodeType which reports what type of node it is. You can see the various types of nodes here (diagram from [MDN](https://developer.mozilla.org/en-US/docs/Web/API/Node/nodeType)):

There are 12 types of nodes, element being one of them



You can see an ELEMENT\_NODE is one particular type of node where the nodeType property has a value of 1.

So document.getElementById("test") can only return one node and it's guaranteed to be an element (a specific type of node). Because of that it just returns the element rather than a list.

Since document.getElementsByClassName("para") can return more than one object, the designers chose to return a nodeList because that's the data type they created for a list of more than one node. Since these can only be elements (only elements typically have a class name), it's technically a nodeList that only has nodes of type element in it and the designers could have made a differently named collection that was an elementList, but they chose to use just one type of collection whether it had only elements in it or not.

**EDIT:** HTML5 defines an HTMLCollection which is a list of HTML Elements (not any node, only Elements). A number of properties or methods in HTML5 now return an HTMLCollection. While it is very similar in interface to a nodeList, a distinction is now made in that it only contains Elements, not any type of node.

The distinction between a nodeList and an HTMLCollection has little impact on how you use one (as far as I can tell), but the designers of HTML5 have now made that distinction.

For example, the element.children property returns a live HTMLCollection.

So all elements are nodes, but not all nodes are elements //comment

# [What’s the difference between nextElementSibling vs nextSibling](https://stackoverflow.com/questions/31097016/whats-the-difference-between-nextelementsibling-vs-nextsibling)?

'nextSibling' returns the next Node object whereas 'nextElementSibling' returns the next Element object, so probably the real question is what is the difference between a Node & an Element?

Basically an Element is specified by an HTML Tag whereas a Node is any Object in the DOM, so an Element is a Node but a Node can also include Text Nodes in the form of whitespace, comments, text characters or line breaks. For more info on Elements vs Nodes see this [Difference between Node object and Element object?](https://stackoverflow.com/questions/9979172/difference-between-node-object-and-element-object)

i.e Take the following DOM snippet

<div id="start"></div>

Me

<p>Hi</p>

Using nextSibling you would get:

console.log(document.getElementById('start').nextSibling); // "\nMe\n"

console.log(document.getElementById('start').nextSibling.nextSibling); // "<p>

Whereas using nextElementSibling you would get:

console.log(document.getElementById('start').nextElementSibling);// "<p>"

Also nextElementSibling is IE10+, being the newer method whereas nextSibling has full browser support.