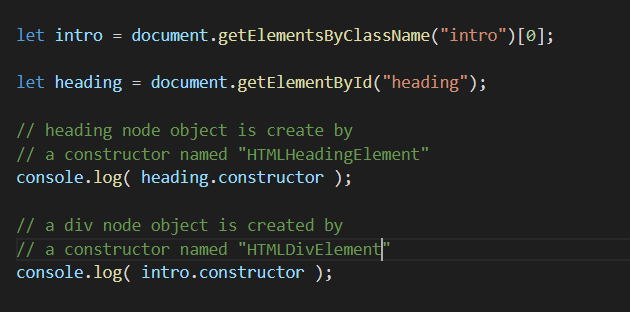


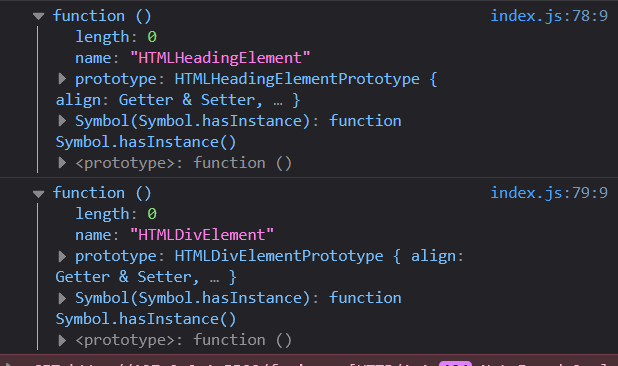
Here is the html format of the document.

Now when browsers parse it, than a DOM tree is created where every HTML elements that we had made in html language now becomes node object.

So what does the parsing of the html format and convert the html elements to the nodes ?

Actually there is constructor in javascript that parse the html written elements and constructs the node object.





TO make DOM tree every html elements needs to be converted to the respective node object.

So there is a constructor for every html element to convert them to the respective node object

**What actually happens in the browser when the HTML document is loaded ?**

So when the HTML document is loaded on the browser all these constructor gets called and all the respective nodes of the HTML Elements gets constructructed. And then DOM tree is constructed.

That DOM tree is live, so we can create and insert any node anywhere on the tree we want

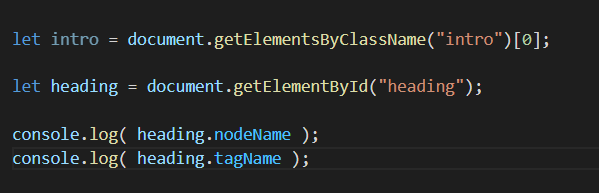
We can delete any node on the tree we want

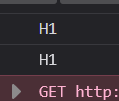
We can replace a node with another node if we want

We can manipulate the DOM tree and it will get updated in real time

**.tagName and .nodeName**

They both gives the same value, so use either of them

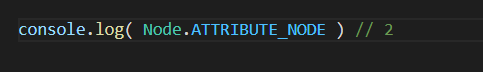




**Working with the attributes**

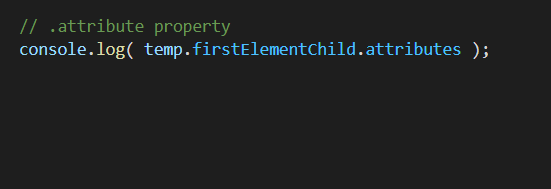
**Attributes are also node object**

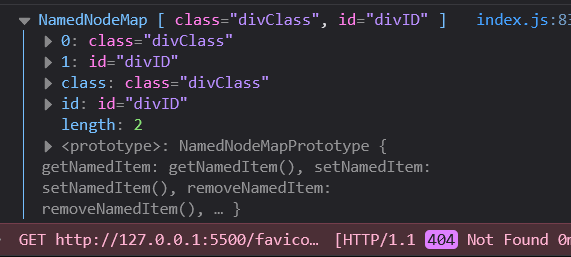
So they also have nodeName, nodeType, nodeValue



**.attributes property**

It gives us the NamedNodeMap





Previously we read about **Nodelist , HTML Collection**

They both were numbered objects, so we could access the items like we do with arrays

This **NamedNodeMap** is also exactly same

In **HTML Collection**, we could access those nodes having id, by directly using their id value, as if that id value was the key of the **HTML Collection object**

Here **NamedNodeMap** is also an object

What is stores are also **node objects of nodeType = Node.ATTRIBUTE\_NODE**

So we can access those **attributes node object** present inside the **NamedNodeMap** directly using the .id and .class also