# Working with the DOM (Browser HTML Code) in Javascript

## DOM

Loaded and rendered HTML code or to be precise the object representation of HTML code that browsers create behind the scene. These are objects which will be exposed to us as javascript objects. These objects are not limited to javascript, in python, some documents can access the DOM objects.

## Window

The document is a property of a window object the window is the root HTML node

## Attributes Vs Properties

**Attributes**: What you add in the HTML tag.

What the browser does with the tag name is create such a DOM object base on the HTML tag and preconfigure some of its properties based on the attributes.

**Properties**: the value stored in the object that is created based on the HTML code.

***Note 1***: Properties name and attributes are not the same all the time, for example, the “id” attribute and property are the same name but the “class” attribute is “className” in the property.

***Note 2***: Some of the properties and attributes have live sync (it means that if we change the property the attribute will change as well), but some don’t have for example “value” attribute in the input tag.

If you do want to change the attribute in the DOM, there is a method for that, “setAttribute”.

***Note:*** Don't forget that JavaScript is a "hosted language". The browser as host environment automatically exposes this DOM API to your JS code.

## Descendants

Direct or indirect child node or element. In this example <p> tag is child and descendant of <div> but <em> tag is only descendant of <div> and not its child:

<div>

<p>

<em>

Test!

</em>

</p>

</div>

In this example <div> is the ancestor of <p> and <em>.

Even spaces in the HTML are considered as a text node in the DOM object created by the browser (For example in the above example space between <p> tag and behind it).

const ul = document.querySelector(“ul”);

**ul. children** only shows HTML nodes but **ul.childNodes** shows all of “**ul**” childes including text nodes for spaces.

**ul. closest(“body”)** to select the nearest ancestor from ul element. inside the parenthesis, you can use a CSS selector.

**ul.previousElementSibling** and **ul.previousSibling** are the same as children and childNodes for selecting siblings.