# 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 which browser create behind the scene. These are objects which will be expose to us as javascript objects. These objects are not limited to javascript, in python there is some documents that can access to the DOM objects.

## Window

Document is a property of window object, actually window is the root html node

## Attributes Vs Properties

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

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

**Properties**: 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 “id” attribute and property are the same name but “class” attribute is “className” in property.

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

If you do want to change 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 exposes this DOM API to your JS code automatically.

## 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 text node in the DOM object created by 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 nearest ancestor from ul element. inside the paranthesis you can use css selector.

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

***Note***: getting siblings is more performance than querySelector.

## Styling DOM Elements

1-Directly target individual CSS styles

2- Controls styles as inline styles on the element;

3- Style property names are based on CSS properties but have adjusted names (e.g. backgroundColor)

1. Via style property

1-Directly set the CSS classes assigned to the element;

2- Set/Control all classes at once;

3- You can also control the id or other properties.

1. Via className

1- Conveniently add, remove or toggle CSS classes;

2- Fine-grained control over classes that are added;

3- Can be used with className (with care).

1. Via classList

The className method will remove all the classes that element has and then replace it with what we added to it.

The classList has methods like: classList.remove , classList.replace, classList.add, classList.toggle.

## Adding Elements via HTML in code

The innerHTML method will replace all the contents. If you want to add another element to old elements:

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

list.innerHTML = list.innerHTML + “<li>Item 4</li>”;

***Note***: The above method has two downsides:

1. It will remove all the elements and then will add it again, so it is not great for performance.
2. If we had an input element by using above method, it will remove what user write in the input.

The better method for that is **insertAdjacentHTML**. This method has some methods:

***beforebegin***

***afterbegin***

***beforeend***

***afterend***

div.insertAdjacentHTML(‘beforeend’,’<p>Something</p>’)