# Document Object Model

Contents

[Document Object Model 1](#_Toc25071695)

[All about the DOM 1](#_Toc25071696)

[Accessing Elements in DOM 1](#_Toc25071697)

[Caching the Selection 1](#_Toc25071698)

[Using Query Selector 2](#_Toc25071699)

[Selecting Multiple Elements 2](#_Toc25071700)

[Traversing the DOM 2](#_Toc25071701)

[HTML VS DOM 3](#_Toc25071702)

## All about the DOM

DOM allows us to access and update webpages in real-time.

DOM allows you to get content.

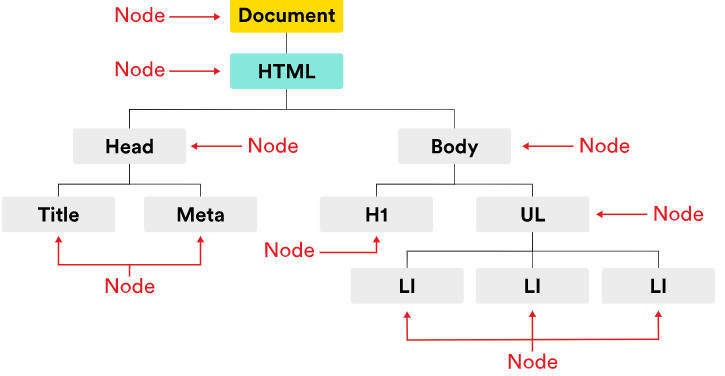
DOM allows you to set content and dynamically update them.

DOM allows you to add animation and effects.

DOM allows you to create event listeners, which allows us to react to a user’s actions.

**The browser pulls in these HTML documents, parses them, and creates object models of the pages in its memory.**

DOM can be expressed as a tree.



You can think of a node as a live object you can access and change using JavaScript.

When the model is updated, those changes are reflected on screen.

DOM make HTML elements come to life!

To inspect and play with the DOM of a webpage in Chrome, right click>Inspect.

Each HTML element is represented by a DOM node.

These nodes are objects that have built-in properties and methods that can be used to access and update that node.

### Accessing Elements in DOM

The syntax is:

document.getElementById(‘main-nav’)

The above statement gets the element wrapped around the below ‘div’.

<div id="main-nav">...</div>

### Caching the Selection

If we’d like to work with a particular element multiple times, we can cache it.

### Using Query Selector

This feature allows us to use the CSS selector (eg. .special ,#specialId) to select an element.

If there are multiple elements which matches the selector, the first element will be picked.

Syntax:

document.querySelector(‘.special’)

document.querySelector(‘#sidebar’)

**document.querySelector(‘h1’)**

**document.querySelector(‘input’)**

### Selecting Multiple Elements

getElementsByClassName(‘vegetarian’)

getElementsByTagName(‘li’)

querySelectorAll(‘.vegetarian’)

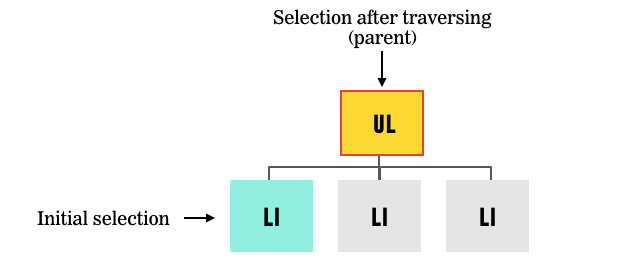
* If the above queries are used, a NodeList will be returned.
* Each individual element can be accessed by referring to its index, for eg. document.getElementsByTagName('li')[2]
* We can also use a loop function to iterate through the different elements.

### Traversing the DOM

Traversing the DOM means moving around different nodes.

A parent node has children node nested within.

By using “document.getElementsByTagName('li')[0].parentNode”,we can move the selection to the parent node of the current selection.



Below are the different functions which can be used to traverse the DOM

* document.getElementsByTagName('li')[0].parentNode
* document.getElementsByTagName('li')[0].nextSibling
* document.getElementsByTagName('li')[0].previousSibling
* document.getElementsByTagName('li')[0].firstChild
* document.getElementsByTagName('li')[0].lastChild

## HTML VS DOM

The DOM is a living model of the page, comprising node objects that can be manipulated with JavaScript. DOM is like the live telecast, reflecting the most updated changes at any point in time. HTML is the static, default set-up of the webpage.

If your HTML isn't properly structured — e.g., you're missing any required elements — the browser will fix its structure as it renders.

And, if you want to use JS to manipulate the DOM (by adding elements, for example), your DOM will render dynamically. On the other hand, your HTML wouldn't reflect these changes as it's static.

