**Document Object and Window Object**

**DOM:**

The Document Object Model (DOM) is the data representation of the objects that comprise the structure and content of a document on the web. DOM is a programming interface. It represents the page so that program can change the document structure, Style and content. The DOM represents the object as nodes and objects, In this way programming languages can interact with the page.

All of the properties, methods and events available for manipulating and creating web pages are organized into objects. For example, the document object that represents the document itself, any table objects that implement the HTML Table Element DOM interface for accessing HTML tables.

**DOM Structure**: The document object provides a hierarchical representation of the HTML elements on a webpage, allowing developers to navigate, modify, and manipulate the content dynamically.

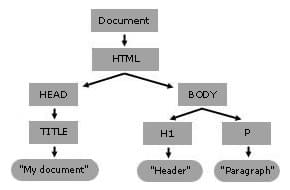
**Methods:** The document object offers methods like getElementById(), getElementsByClassName(), and querySelector(), enabling developers to select specific elements based on their IDs, classes, or other attributes.

**Properties:** Properties such as document.title, document.URL, and document.body provide information about the document's title, URL, and body, respectively.

**Content Manipulation:** Through the document object, you can dynamically create, modify, or delete elements, attributes, and text content on a webpage.

**DOM Tree:**

A DOM tree is a tree structure whose nodes represent an HTML or XML document's contents. Each HTML or XML document has a DOM tree representation.



**Example program for above DOM tree:**

***HTML:***

<html lang="en">

<head>

<title>Document Object Model</title>

</head>

<body>

<input type="button" value="Change this document." onclick="change()" />

<h1>Header</h1>

<p>Paragraph</p>

</body>

</html>

**JavaScript:**

Function change()

{

const header = document.getElementByTagName(“h2”).item(0)

header.firstChild.data = “A Dynamic Document”

const para = document.getElementByTagName(“p”).item(0)

para.firstChild.data = “This is first paragraph”

const newText = document.createTextNode(“This is second Para”)

const newElement = document.createElement(“p”)

newElement.appendChild(newText)

para.parentNode.appendChild(newElement)

}

We have used several DOM methods in JavaScript. By using DOM methods we can build several things in HTML document. DOM is mainly used to manipulate the HTML document.

**Window Object**

The Window object represents the window that contains a DOM document.The Window object provides access to various properties and methods that enable interaction with the browser environment, including manipulating the document, handling events, managing timers, displaying dialog boxes, and more.

In a tabbed browser, each tab is represented by its own Window object. The global window seen by JavaScript code running within a given tab always represents the tab in which the code is running.

The global window seen by JavaScript code running within a given tab always represents the tab in which the code is running. Even in a tabbed browser, some properties and methods still apply to the overall window that contains the tab, such as resizeTo() and innerHeight. Generally, anything that can't reasonably pertain to a tab pertains to the window instead.

**Global Scope:** The window object serves as the global scope in JavaScript, meaning variables and functions declared without the var, let, or const keyword become properties of the window object.

**Browser Information:** The window object offers properties like window.innerWidth, window.innerHeight, and window.navigator to provide information about the browser's dimensions, navigator object, and other related details.

**Navigation:** Methods such as window.open(), window.close(), and window.location allow developers to control browser navigation, open new windows or tabs, and manipulate the current URL.

**Timers:** The window object provides functions like setTimeout(), setInterval(), and clearTimeout() to manage time-based operations and execute code asynchronously.

**Window Object Properties:**

Window interface inherits the EventTarget interface.

Some of the Window Properties are below.

**Window.caches:**

Returns the CacheStorage object associated with the current context. This object enables functionality such as storing assets for offline use and generating custom responses to requests.

**Window.closed:**

This property indicates whether the current window is closed or not.

**Window.cookieStore:**

Returns a reference to the CookieStore object for the current document context.

**Window.credentialless:**

Returns a boolean that indicates whether the current document was loaded inside a credentialless<iframe>.

**Window Object Methods:**

Some of the Window object methods are below.

**Window.atob():**

Decodes a string of data which has been encoded using base-64 encoding.

**Window.alert():**

Displays an alert dialog.

**Window.close():**

Closes the current window.

**Window.confirm():**

Displays a dialog with a message that the user needs to respond to.

**Window.find():**

Searches for a given string in a window.

**Some of the Window object events are:**

We can listen to these events using addEventListener().

**error:**

This event is fired when a resource failed to load, or can't be used.

**storage:**

This is fired when storage area has been modified in the context of another document.

**languagechange:**

This event is fired when user preferred language change.

**Conclusion:**

while the document object focuses on representing and manipulating the content within an HTML document, the window object provides a broader scope, encompassing the browser window's properties, methods, and functionalities.