**Difference between Window, Document and Screen in JavaScript**

Window object and Document object often look alike and confusing ??

A clear understanding of Browser Object Model (BOM) and Document Object Model(DOM) resolves the problem.

## Browser Object Model (BOM)

The Browser Object Model contains objects that represent the current browser window or tab. The topmost object in the BOM is the **window** object representing the window or tab or an iframe sometimes. Window object has properties like browser history, location and the device’s screen etc. In case of multi tab browser, a window object represents a single tab, but some of its properties like innerHeight, innerWidthand methods like resizeTo() will affect the whole browser window.

## **Document Object Model**

When a web page is loaded, the browser creates a Document Object Model of the page. The document object represents the whole html document as a tree of Objects(HTML, HEAD, BODY, and other HTML tags). It is the root element that represents the html document.

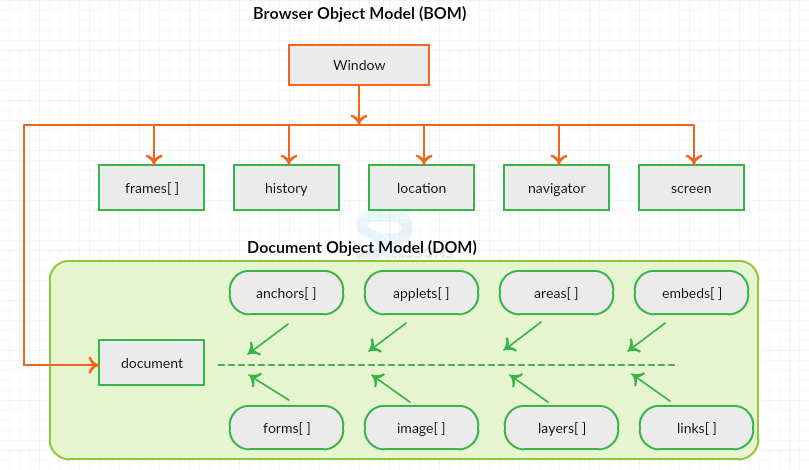


Figure 1 : Hierarchy of Window and Document object using BOM and DOM

Now lets visualize the difference between window and document.

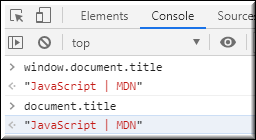
## Window Vs Document

**Window object** : It is the top most object and outermost element of the object hierarchy as shown in Figure 1.

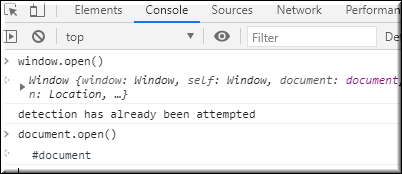
**Document object** : Each HTML document that gets loaded into a window becomes a document object. The document contains the contents of the page. Using document object, JavaScript can modify, add and delete the HTML elements, attributes CSS styles in the page

The window object represents a window/tab containing a DOM document where as document object is property of window object that points to the DOM document loaded in that window.

You can access a document object either using window.document property or using document object directly as window is global object. In the below example, title is the property of document object.



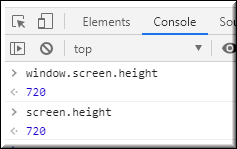
The other major difference is that both window object and document object have properties and methods. Few method names are same in both objects but with different behavior. In the below example window.open() opens a new tab or window and document.open() creates a blank document within the window.



## Screen

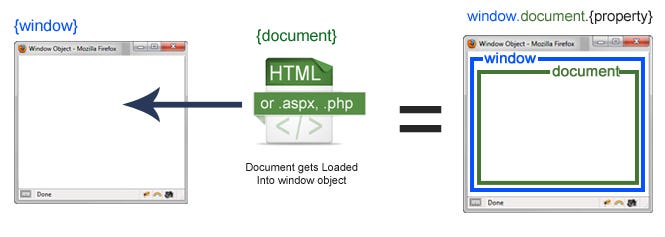
Screen is the window property that holds information of browser screen. It refers to screen object associated with that window **object**. Used to display screen width, height, colorDepth, pixelDepth etc

Similar to document screen can be accessed either by window.screen or screen object directly. Screen object doesn't have any methods as in window and document objects.



# WINDOW

**Window** is the main JavaScript object root, aka the **global object** in a browser, also can be treated as the root of the document object model. You can access it as **window.**



Well, the window is the first thing that gets loaded into the browser. This window object has the majority of the properties like length, innerWidth, innerHeight, name, if it has been closed, its parents, and more.

# DOCUMENT

What about the **document** object then? The Document object(**DOM**) is your html, aspx, php, or other document that will be loaded into the browser. The document actually gets loaded inside the window object and has properties available to it like title, URL, cookie, etc. What does this really mean? That means if you want to access a property for the window it is window.property, if it is document it is window.document.property which is also available in short as document.property.

**window.document** or just **document** is the main object of the potentially visible (or better yet: rendered) document object model/DOM.

Since **window** is the global object you can reference any properties of it with just the property name - so you do not have to write down **window**. - it will be figured out by the runtime.

# ****SCREEN****

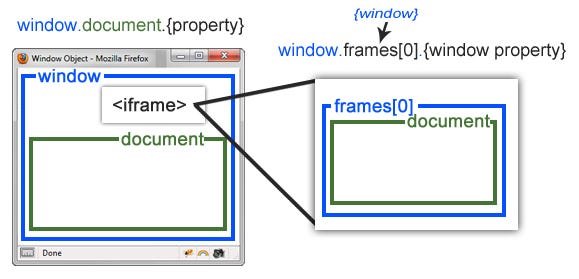
The **Window** object also has a **screen** object with properties describing the physical display:

* screen properties width and height are the full screen
* screen properties availWidth and availHeight omit the toolbar

**window.screen** or just **screen** is a small information object about physical screen dimensions.

## Note:

But what happens once an **IFRAME** is introduced? Uh oh… frameage.



An iframe actually is considered as a new window with its own document loaded into it. Here is where it may seem a little odd, but does make sense if you think about it. The original, parent window, is responsible for other windows to be loaded, not the document.

The property to access a frame is window.frames[], which is an array of all the frames. If you only have one iframe you access it by using window.frames[0]. Since the iframe is also a window object, accessing window pro

perties of that frame is done by using window.frames[0].mywindowproperty.