- ***JAVASCRIPT WEBAPI AND INTERFACES***
    - 1.JavaScript Browser Object
    - 2.JavaScript Window Object
    - 3.JavaScript History Object
    - 4.JavaScript Navigator Object
    - 5.JavaScript Location Object
    - 6.JavaScript Screen Object
    - 7.JavaScript Document Object

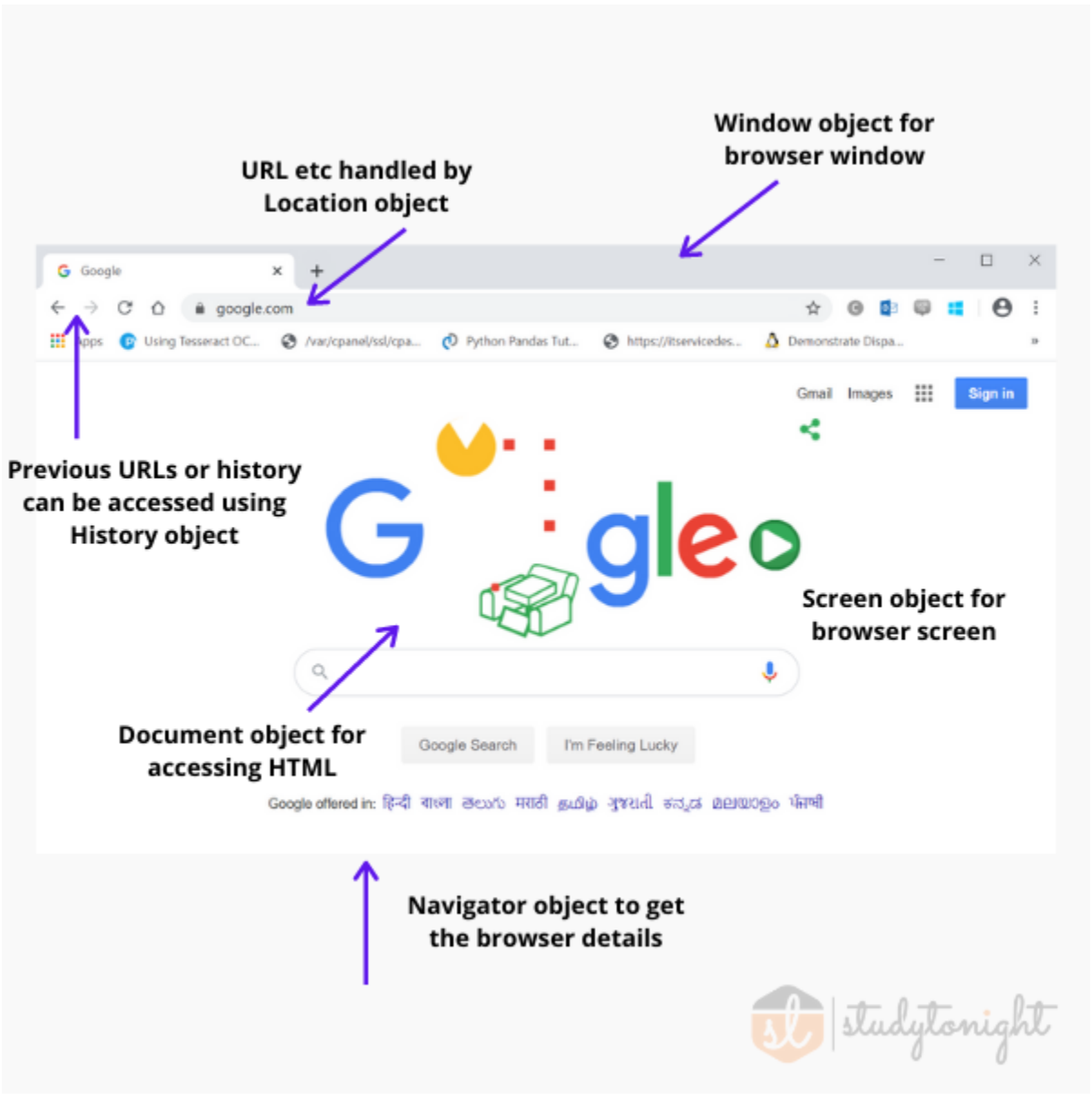
1.JavaScript Browser Object Model

JavaScript provides WebAPIs and Interfaces(object types) that we can use while developing web application or website. These APIs and objects help us in controlling the lifecycle of the webpage and performing various actions like getting browser information, managing screen size, opening and closing new browser window, getting URL information or updating URL, getting cookies and local storage, etc.

There is so much that you can do using these WebAPIs and Interfaces. All the supported Web APIs and Interfaces in JavaScript are listed on the Mozilla Developer website.

The Iterfaces (object types) which help us **interact with the browser window** are known as **Browser objects**. Browser object is not an official term but it's a group of objects which belongs to different WebAPIs but are used for managing various browser related information and actions.

For example, when an HTML document is opened in a browser window, the browser interprets the document as a collection of hierarchical objects(HTML tags) and accordingly displays the data contained in these objects(HTML page rendering). The browser parses the document and creates a collection of objects, which defines the documents and its details. We have shown the various objects that can be used to access various components of the browser window in the picture below:



The browser objects are of various types, used for interacting with the browser and belongs to different APIs. The collection of these Browser objects is also known as **Browser object Model**(BOM).

The default object of browser is Window which means you can call its functions directly.

Browser Objects:

The objects listed below are called browser objects.

- Window – part of DOM API
- Navigator
- Document – part of DOM API
- Screen – property of Window object
- History – property of Window object
- Location – property of Window and Document object

Window Object

It is used to interact with the browser window which displays the web page. It generally **represents a tab in browser**, but some actions like window width and height will affect the complete browser window.

We have covered JavaScript Window Object separately, in detail.

Navigator Object

It acts as a storehouse of all the data and **information about the Browser software used to access the webpage** and this object is used to fetch information related to the browser for example, whether the user is using Chrome browser or Safari browser, which version of browser is being used etc.

We have covered JavaScript Navigator Object separately, in detail.

Document Object

This object **represent the HTML document** that is loaded into the browser. A document object is an object that provides access to all HTML elements of a document(webpage). We can use this object to append a new HTML tag to the webpage, modify any exsiting HTML tag, etc.

We have covered JavaScript Document Object separately, in detail.

History Object

It stores Uniform Resource Locator(URLs) visited by a user in the browser. It is a built-in object which is used to get browser history. This object is a property of the JavaScript Window object.

We have covered JavaScript History Object separately, in detail.

Screen Object

It is a built-in object which is used to **fetch information related to the browser screen**, like the screen size, etc. It is also obtained from the Window object.

We have covered JavaScript Screen Object separately, in detail.

Location Object

Location is a built-in object which **represent the location of the object to which it is linked**, which can be Window or Document. Both the Document and Window interface have a linked location property.

We have covered JavaScript Location Object separately, in detail.

From the next tutorial, we will cover all these objects, one by one, in detail, with examples to see how we can use them to interact with browser and perform various different actions.

2.JavaScript Window Object

JavaScript Window is a **global Interface (object type)** which is used to **control the browser window lifecycle** and perform various operations on it.

A global variable window, which represents the current browser window in which the code is running, is available in our JavaScript code and can be directly accessed by using `window` literal.

It represents a window containing a webpage which in turn is represented by the document object. A window can be a new window, a new tab, a frame set or individual frame created with JavaScript.

In case of multi tab browser, a **window object represents a single tab**, but some of its properties like `innerHeight`, `innerWidth` and methods like `resizeTo()` will affect the whole browser window.

Whenever you open a new window or tab, a window object representing that window/tab is automatically created.

The properties of a window object are used to retrieve the information about the window that is currently open, whereas its methods are used to perform specific tasks like opening, maximizing, minimizing the window, etc.

To understand the window object, let's use it to perform some operations and see how it work.

Using JavaScript Window Object

Let's use the JavaScript window object to create a new window, using the `open()` method. This method creates a new window and returns an object which further can be used to manage that window.

JS OPEN A NEW WINDOW EXAMPLE

_Run

```
<!doctype html>
      <head>
            <title>Open a new Window</title>
      </head>
      <body>
            <h3>Click to open a new Window:</h3>

        <button onclick="createWindow()">Open a Window</button>
        <p id="result"></p>

        <script>
        function createWindow() {
            let url = "https://studytonight.com";
            let win = window.open(url, "My New Window", "width=300, height=200");
            document.getElementById("result").innerHTML = win.name + " - " +
win.opener.location;
        }
        </script>

      </body>
</html>
```

In the code above, we have used the window object of the existing window to create a new window using the `open()` method. In the `open()` method we can provide the URL to be opened in the new window(we can keep it blank as well), **name of the window**, the **width** and the **height** of the window to be created.

Following is the syntax for the window object open() method:

```
let newWindow = window.open(url, windowName, [windowFeatures]);
```

Copy

We can provide, as many properties as we want, while creating a new window.

When the `open()` method is executed, it returns the reference of the window object for the new window created, which you can assign to a variable, like we have done in the code above. We have assigned the value returned by the `window.open()` method to the `win` variable.

The we have used the `win` variable to access the new window, like getting the name of the window, getting location of the window which opened the new window etc.

There are many properties and methods for the window object, which we have listed down below.

Find Dimensions of a Window

We can get the height and width of a window by using the window object built-in properties.

ADD CONTENT TO WINDOW EXAMPLE

_Run

```
<!doctype html>
```

```
        <head>
              <title>Add content to Window</title>
        </head>
        <body>
          <script>
            let h = window.outerHeight
            let w = window.outerWidth
            window.document.write("Height and width of the window are: "+h+" and "+w)
          </script>

        </body>
</html>
```

We can also access the document object using a window object(window.document) which gives us access to the HTML document, hence we can add new HTML element, or write any content to the document, like we did in the example above.

JavaScript Window Object Properties

The wondow object properties refer to the variables created inside the windows object.

In JavaScript, all the available data is attached to the window object as properties.

We can access window object's properties like: window.propertyname where **propertyname** is the name of property.

A table of most popular window object properties is given below:

Property	Description
closed	returns a boolean value that specifies whether a window has been closed or not
document	specifies a document object in the window.
history	specifies a history object for the window.
frames	specifies an array of all the frames in the current window
defaultStatus	specifies the default message that has to be appeared in the status bar of Window.
innerHeight	specifies the inner height of the window's content area.
innerWidth	specifies the inner width of the window's content area.
length	specifies the number of frames contained in the window.

Property	Description
location	specifies the location object for window
name	specifies the name for the window
top	specifies the reference of the topmost browser window.
self	returns the reference of current active frame or window.
parent	returns the parent frame or window of current window.
status	specifies the message that is displayed in the status bar of the window when an activity is performed on the Window.
screenleft	specifies the x coordinate of window relative to a user's monitor screen
screenTop	Specifies the y coordinate of window relative to a user's monitor screen
screenX	specifies the x coordinate for window relative to a user's monitor screen
screenY	Specifies the y coordinate for window relative to a user's monitor screen

Let's take an example to see a few of these properties in action:

JS WINDOW OBJECT PROPERTY

_Run

```
<!doctype html>
	<head>
		<title>JS Window object properties</title>
	</head>
	<body>
		<h3>Create a Window by clicking</h3>

	  <button onclick="createWindow()">Open a Window</button>

	  <script>
		function createWindow() {
			var win = window.open("", "My Window", "width=500,
height=200,screenX=100,screenY=100");
			// window properties
			var isclose = win.closed;
			var name = win.name;
			// writing in the current document
```

```
                    document.write(isclose +"<br>");
                    document.write(name +"<br>");
                    document.write(win.screenY +"<br>");
                    document.write(win.screenX +"<br>");
                    // we can access the new window document like this
                    win.document.write("Hello World!");
                }
            </script>

        </body>
</html>
```

In the example above, we have created a new window, just to make you familiar with new window creation and also, to make you understand that when we create a new window, then the current window has a different window object and the new window will have a separate window object.

We have then tried to access some properties of the new window created.

TASK: Try some more windows property in the above code playground to practice and understant how window properties work.

JavaScript Window Object Methods

The window object methods refer to the functions created inside the Window Object, which can be used to perform various actions on the browser window, such as how it displays a message or gets input from the user.

Below we have listed down some of the most commonly used window object methods:

Method	Description
alert()	specifies a method that displays an alert box with a message an OK button.
blur()	specifies a method that removes the focus from the current window.
clearInterval()	specifies a method that clears the timer, which is set by using setInterval() method.
close()	specifies a method that closes the current window.
confirm()	specifies a method that displays a dialogue box with a message and two buttons OK and Cancel
focus()	specifies a method that sets the focus on current window.
open()	specifies a method that opens a new browser window

Method	Description
moveTo()	specifies a method that moves a window to a specified position
moveBy()	specifies a Method that moves a window relative to its current position.
prompt()	specifies a method that prompts for input.
print()	specifies a method that sends a print command to print the content of the current window.
setTimeout()	specifies a method that evaluates an expression after a specified number of milliseconds.
setInterval()	specifies a method that evaluates an expression at specified time intervals (in milliseconds)
resizeBy()	specifies the amount by which the window will be resized
resizeTo()	used for dynamically resizing the window
scroll()	scrolls the window to a particular place in document
scrollBy()	scrolls the window by the given value
stop()	this method stops window loading

Let's take an example and see a few of the above methods in action:

JS WINDOW OBJECT PROPERTY

Run

```
<!doctype html>
	<head>
		<title>JS Window object properties</title>
	</head>
	<body>
		<h3>Create a Window by clicking</h3>

	  <button onclick="createWindow()">Open a Window</button>

	  <script>
		function createWindow() {
			var win = window.open("", "My Window", "width=500,
height=200,screenX=100,screenY=100");
```

```
                    // window properties
                    var isclose = win.closed;
                    var name = win.name;
                    // writing in the current document
                    document.write(isclose +"<br>");
                    document.write(name +"<br>");
                    document.write(win.screenY +"<br>");
                    document.write(win.screenX +"<br>");
                    // we can access the new window document like this
                    win.document.write("Hello World!");
                }
            </script>

        </body>
</html>
```

In the example above, we have used some window methods, you can try more methods. In the next tutorial we will learn about the **History** object, which is a property of the window object.

3.JavaScript History Object

JavaScript History is a built-in Interface (object type) which is used to access the browser window session history. It contains the list of URLs visted by a user in a particular browser window, tab or frame, represented by the window object.

The history object is a property of the JavaScript window object, and can be accessed through the `window.history` property, and is used to access the session history for that window object.

It provides useful methods and properties that let us navigate back and forth through the window's session history.

Let's take an example to see how this works to understand it better.

JavaScript History Object Example

The History object's `back()` method is used to go back to the previous URL accessed in the browser window, in the history list.

It is same as clicking the **"Back button"** in your browser.

```
<button onclick="goBack()">Go Back</button>



<script>

        function goback(){

            window.history.back();

        }

</script>
```

Copy

Similarly, we can use the history object's `forward()` method to load the next URL in the history list. If next URL is not present, it does nothing.

It is same as clicking the **"Forward button"** in your browser.

```
<button onclick="goforward()">Go Forward</button>



<script>

        function goforward(){

            window.history.forward();
```

```
      }

</script>
```

Copy

Apart from the `back()` and `forward()`, JavaScript history object provides more methods like `go()` which can be used to **jump to any specific URL** present in the history object.

To **move to any specific page**, we can use the `go()` method, it takes an integer argument that works as an **index value for history URL list**. See the below example:

```
<script>

    // move to one page backward

       window.history.go(-1);



    // move to two page backward

    window.history.go(-2);



    // move to one page forward

    window.history.go(1);



    // move to two page forward

    window.history.go(2);

</script>
```

Copy

We can also use the `go()` method of history object **to refresh the current page**. In the example below, we have called the `go()` method two times and it works the same in the both scenario. We can call it anytime when we want to refresh/reload the current page.

```
<script>

// Refresh the page

        window.history.go(0);



        window.history.go();



</script>
```

Copy

JavaScript History Object Property

Following are the properties of the History object. The `length` property is used the most.

Properties	Description
length	specifies the number of elements contained in the object
scrollRestoration	allows web applications to explicitly set default scroll restoration behavior on history navigation
state	represents the state at the top of the history stack

Let's take an example to get number of pages visited by a user in a browser window. We will use the `length` property that contains information for the visited pages. See the below example:

JS HISTORY LENGTH EXAMPLE

Run

```
<!doctype html>
      <head>
            <title>JS History Length Example</title>
      </head>
      <body>
            <h3>History Length Property</h3>
            <script>
           // fetch number of visited pages
           let len = window.history.length
           document.write("You visited " +len+ " Pages.")
        </script>
      </body>
</html>
```

JavaScript History Object Methods

JavaScript History object methods refer to the functions created inside the history object. These methods specify the actions related to the URLs visited in the Browser window's current session.

Methods	Description
back()	specifies a method that loads the previous URL from the history list.
forward()	specifies a method that loads the next URL from the history list.
go()	specifies a method that loads a specific URL from the history list.
pushState()	used to push the given data onto the session history stack with the specified title

Methods	Description
replaceState()	used to update the most recent entry on the history stack to the specified data, title, and, if provided, URL

We have explained each method with examples in the beginning of this tutorial itself.

In the next tutorial we will learn about the Navigator browser object.

4.JavaScript Navigator Object

JavaScript Navigator Object is used to fetch information related to the browser(useragent), like the browser name, browser version, operating system information, etc.

JavaScript Navigator object is also a property of the JavaScript Window object and can be accessed using the read only property window.navigator for the current browser window.

In JavaScript, the Navigator Object includes some properties and methods which help us to navigate from one element to another.

Navigator Object Properties

The properties of the navigator object are the variables created inside the Navigator Object.

We can access Navigator Object Property as: **navigator.propertyname** where **propertyname** is the name of property.

Properties	Description
appcodename	specifies the code name of the browser (*experimental property - can return incorrect value*)
appname	specifies the name of the browser (*experimental property - can return incorrect value*)
appversion	specifies the version of browser being used (*experimental property - can return incorrect value*)
cookieEnabled	specifies whether cookies are enabled or not in the browser
platform	contains a string indicating the machine type for which the browser was compiled.
useragent	contains a string representing the value of **user-agent header** sent by the client to server in HTTP protocol

Properties	Description
geolocation	returns object of GeoLocation which can be used to get the location information of the device.
onLine	specifies whether the browser is online or not.
language	returns a string with the preferred browser language, for example, **en-US** for English.
languages	returns a string with all the languages supported by the browser in order of user preference.

There are some more expreimental properties available in the Navigator object, but the ones mentioned above are most commonly used properties.

Let's see a code example with some of these properties in action.

JS NAVIGATOR OBJECT PROPERTIES

_Run

```
<!doctype html>
      <head>
            <title>JS Navigator Object Properties</title>
      </head>
      <body>
            <h3>Navigator Object Example</h3>

        <script>
            let appc =  navigator.appCodeName;
            let appn =  navigator.appName;
            let appv =  navigator.appVersion;
            let appco = navigator.cookieEnabled;
            let lan   = navigator.language;
            let onl = navigator.onLine;
            let pla = navigator.platform;
            let usra = navigator.userAgent;

            document.write(appc +"<br>");
            document.write(appn +"<br>");
            document.write(appv +"<br>");
            document.write(appco +"<br>");
            document.write(lan +"<br>");
            document.write(onl +"<br>");
            document.write(pla +"<br>");
            document.write(usra +"<br>");
        </script>

      </body>
</html>
```

Navigator Object Methods

The Navigator object doesn't have many standardised methods, most of the methods available are experimental.

Method Name	Description
registerProtocolHandler()	allows websites to register themselves as protocol handler so that they are allowed to open some standard URL schemes like **mailto:**, **tel:**, **sms:**, etc,
share()	used to utilize the native support of sharing available in browser.
sendBeacon()	used to send a small data packet from the client to the server
vibrate()	if supported, this can be used to make the device vibrate, if not supported, nothing happens. This is useful in mobile browsers.

Use of Navigator Object:

The navigator object can be used for multiple specific usecases which can help you make your website better, like:

1. Making you web app or website more **compatible with different browsers** by writing specific code for different browsers to resolve compatibility issues.
2. You can check if the browser is online or not, to show **some message if the browser is disconnected** from internet.
3. You can use the location information of the user to **show location specific content**.
4. You can **set the preferred language** as the website language if you support multiple languages on your website.

All the points mentioned above can improve the user experience of your website multifold.

5.JavaScript Location Object

JavaScript Location is a built-in Interface (object type) which represents the location (URL) of the object to which it is attached. Both the Window object and the Document object have this property. The window objects location property gives us access to the browser window's location whereas the document location property gives us the location object for a particular document.

It is used to fetch information of the current URL. The Location object allows you to navigate to another webpage as well.

Suppose we want to get the current website's hostname then we can use the hostname property of the location object. See the below example:

OCATION OBJECT BASIC EXAMPLE

Run

```
<!doctype html>
	<head>
		<title>Location object Basic Example</title>
	</head>
	<body>
		<p>Get Website Host name</p>

		<script>
			var x = location.hostname;
		document.write(x);

		</script>
	</body>
</html>
```

There are many different properties of the location object and many methods too.

JavaScript Location Object Properties

JavaScript Location Object has many properties using which we can access the various components of a webpage's URL and even change it.

Here are some commonly used properties for the location object:

Property	Description
href	Represents a string specifying the entire URL, for instance http://www.javascriptstudytonight.com:80/order.cgi?batch=1#intro
protocol	Represents a String at the begining of a URL to the first colon(:),which specifies the Method of access to the URL , for instance http: or https:
host	Represents a String consisting of hostname and port Strings,for instance:- www.javascriptstudytonight.com:80
hostname	Represents the server name, subdomain, and domain name (or IP address)of a URL, for instance www.javascriptstudytonight.com
port	Represents a string specifying the communication port that the server uses, for instance 80
pathname	Represents a String Portion of a URL, specifying how a particular resource can be accessed, for instance: order.cgi
search	Represents a string beginning with a question mark that specifies any query information in an HTTP URL, for instance batch=1
hash	Represents a string beginning with a hash(#), which specifies an anchor name in an HTTP URL, for instance #intro

Let's take an example and get to know about the properties of Location object.

JS LOCATION OBJECT PROPERTIES EXAMPLE

Run

```
<!doctype html>
	<head>
		<title>Location object Properties</title>
	</head>
	<body>
		<h3>Location Properties</h3>

		<script>
		    // Hostname
```

```
                let x = location.hostname;
            document.write(x +"<br>");
            // href
            x = location.href;
            document.write(x +"<br>");
            // protocol
            x = location.protocol;
            document.write(x +"<br>");
            // host
            x = location.host;
            document.write(x +"<br>");
            // pathname
            x = location.pathname;
            document.write(x +"<br>");
             </script>
      </body>
</html>
```

Now let's see the methods of the location object.

JavaScript Location Object Methods

Location Object methods refers to the functions created inside the location interface which can be used to perform various operations on the URL like reload it, change it, etc.

Method	Description
assign()	Loads a new Document in the Browser
reload()	Reloads the current document that is contained in location.href property.
replace()	Replaces the current document with the specified new one. In addition, you cannot navigate back to the previous document using the Browser's back button.

To load a new document, We can use the assign() method of Location object. But in this example, we have used the href property and the replace() methods too. They all can be used to load a document. See the below example:

LOCATION OBJECT METHOD EXAMPLE

Run

```
<!doctype html>
      <head>
            <title>Location object Method example</title>
      </head>
      <body>
            <p>Location Methods example</p>

        <button onclick="load1()">visit studytonight.com</button>
        <button onclick="load2()">visit studytonight.com</button>
        <button onclick="load3()">visit studytonight.com</button>

            <script>
                // assign method
                function load1(){
                      location.assign("https://www.studytonight.com");
                }

                // href
                function load2(){
                    location.href="https://www.studytonight.com";
                }

                // replace()
```

```
                function load3(){
                    location.replace("https://www.studytonight.com");
                }

            </script>
        </body>
</html>
```

6.JavaScript Screen Object

JavaScript Screen is a built-in Interface (object type) that is used to fetch information related to the browser screen on which the current window is rendered.

It provides information about the dimensions of the display screen such as its height, width, color bits, etc.

Since the window object is at the top of the scope chain, the property window.screen gives the Screen object, but the screen object can be accessed without specifying the window too, in that case JavaScript automatically detects the browser window, which is created by JavaScript Runtime Engine.

Let's take a simple example to see the working of the screen object.

Find the Height and Width of Screen:

In the example below we have used the screen object to get the width and height of the screen:

JS SCREEN OBJECT EXAMPLE

Run

```
<!doctype html>
        <head>
                <title>JS Screen Object Example</title>
        </head>
        <body>
                <h3>JavaScript Screen Object</h3>

                <script>
                        document.writeln("Total Height: " + screen.height +"<br>");
                document.writeln("Total Width: " + screen.width +"<br>");
                </script>
        </body>
</html>
```

JavaScript Screen Object Properties

JavaScript Screen object includes some properties that are used to get browser screen information. A table of such properties is given below.

Property	Description
availHeight	specifies the height of screen, excluding the windows taskbar
availWidth	specifies the width of the screen, excluding the Windows taskbar
colorDepth	specifies the depth of color palette, in bits, to display images

Property	Description
height	specifies the total height of the screen
pixelDepth	specifies the color Resolution, in bits per pixel, of the screen
width	specifies the total width of the screen

Now, we will take an example to clear the concepts of Screen Object Properties:

JS SCREEN OBJECT PROPERTIES EXAMPLE

_Run

```
<!DOCTYPE HTML>
<body>
    <script Language="javascript">
        document.writeln("Total Height : " +screen.height+"<br/>");
        document.writeln("Total Width : " +screen.width+"<br/>");
        document.writeln("Available Width : " +screen.availWidth + "<br/>");
        document.writeln("Available Height  : " +screen.availHeight + "<br/>");
        document.writeln("Screen Color Depth : " +screen.colorDepth + "<br/>");
        document.writeln("Screen Pixel Depth  : " + screen.pixelDepth + "<br/>");
    </script>
</body>
</htm
```

The Screen object **colorDepth** property is the number of bits used to represent the color of a single pixel. It returns the bit depth of the color palette for displaying images.

JavaScript Screen Object Methods

There are no direct methods in the Screen method object.

Uses of Screen Object

The JavaScript Screen object can be used for improving the user experience by controlling the UI of the web app or website based on the screen size.

7.JavaScript Document Object

JavaScript Document object is an object that provides access to all HTML elements of a document. When an HTML document is loaded into a browser window, then it becomes a document object.

The document object stores the elements of an HTML document, such as HTML, HEAD, BODY, and other HTML tags as objects.

A document object is a child object of the Window object, which refers to the browser.

You can access a document object either using `window.document` property or using object directly.

See, What we can do with Document Object:

- Suppose you have created a FORM in an HTML document using the FORM element and added some text fields and Buttons on the Form. On Submitting the Form you want to validate the input or display input on another page. In this situation, you can use a document object which is a child object of the window object. Using the document object, you can access the elements of the form and validate the Input.
- The Document Object provides different collection elements, such as anchor and Links which helps you to count the number of specific elements on a form.
- The Document Object also provides various properties such as `URL`, `bgcolor`, etc. which will allow you to retrieve the details of a document and various methods such as `open()`, `close()`, `write()`, `getElementById()`, `getElementByName()`, etc.

which allow you to perform various tasks like opening an HTML document to display output and writing a text on Web Page.
- The Document Object also allows you to create cookies for a webpage by providing a property named cookie. A cookie stores information about the user's computer, which helps in accessing visited websites.

Now let's explore document object methods and properties.

JavaScript Document Object Properties

As we know, a property of an object is the value associated with the object. The property is accessed by using the following notation:

```
objectName.propertyName
```

Copy

where **objectName** is the name of the object and **propertyName** is the name of its property.

Now we will show you the properties of document object in the table given below:

Properties	Description
cookie	returns a report that contains all the visible and unexpired cookies associated with the document
domain	returns the domain name of the server from which the document has originated
lastModified	returns the date on which document was last modified
documentMode	returns the mode used by the browser to process the document
readyState	returns the loading status of the document.
referrer	returns the URL of the documents referred to in an HTML document
title	returns the name of the HTML document defined between the starting and ending tags of the TITLE element
URL	returns the full URL of the HTML document.

Let's take an example, to see the document object properties in action:

DOCUMENT-PROPERTY

Run

```
<!doctype html>
      <head>
            <style>
                  /* CSS comes here */
```

```
            </style>
            <title>Document Properties</title>
      </head>
      <body>
            <h3>Document Properties Example</h3>

            <script>

                  document.write(document.domain +"<br>")
                  document.write(document.lastModified +"<br>")
                  document.write(document.documentMode +"<br>")
                  document.write(document.title +"<br>")
                  document.write(document.url +"<br>")

            </script>
      </body>
</html>
```

JavaScript Document Object Methods

JavaScript Document object also provides various methods to access HTML elements.

Now we will show you some of the commonly used methods of the document object:

Methods	Description	Syntax
open()	opens an HTML document to display the output	`document.open(mimetype, replace)` Copy
close()	closes an HTML document	`document.close()` Copy
write()	Writes HTML expressions or JavaScript code into an HTML document	`document.write(exp1, exp2, ...)` Copy
writeln()	write a new line character after each HTML expressions or JavaScript Code	`document.writeln(exp1, exp2, ...)` Copy
getElementById()	returns the reference of first element of an HTML document with the specified id.	`document.getElementById(id)` Copy
getElementByName()	returns the reference of an element with the specified name	`document.getElementsByName(name)` Copy
getElementsByTagName()	returns all elements with the specified tagname	`document.getElementsByTagName(tagname)`

Methods	Description	Syntax
		Copy

Now let's take an example, to see the document object methods like `open()`, `write()`, and `getElementById()` to explain their usage. See the below example:

JS DOCUMENT METHOD EXAMPLE

Run

```
<!doctype html>
      <head>
            <title>JS Document Methods Example</title>
      </head>
      <body>
            <h3>Document Methods Example</h3>
            <p id="demo">It Will change</p>

            <script>
                // Open document
                document.open();
                // writing
                   document.write("Hello" +"<br>");
                   document.getElementById("demo").innerHTML = "Set by ID";
            </script>
      </body>
</html>
```

Also Read :- How to Build a Discord Bot using JavaScript