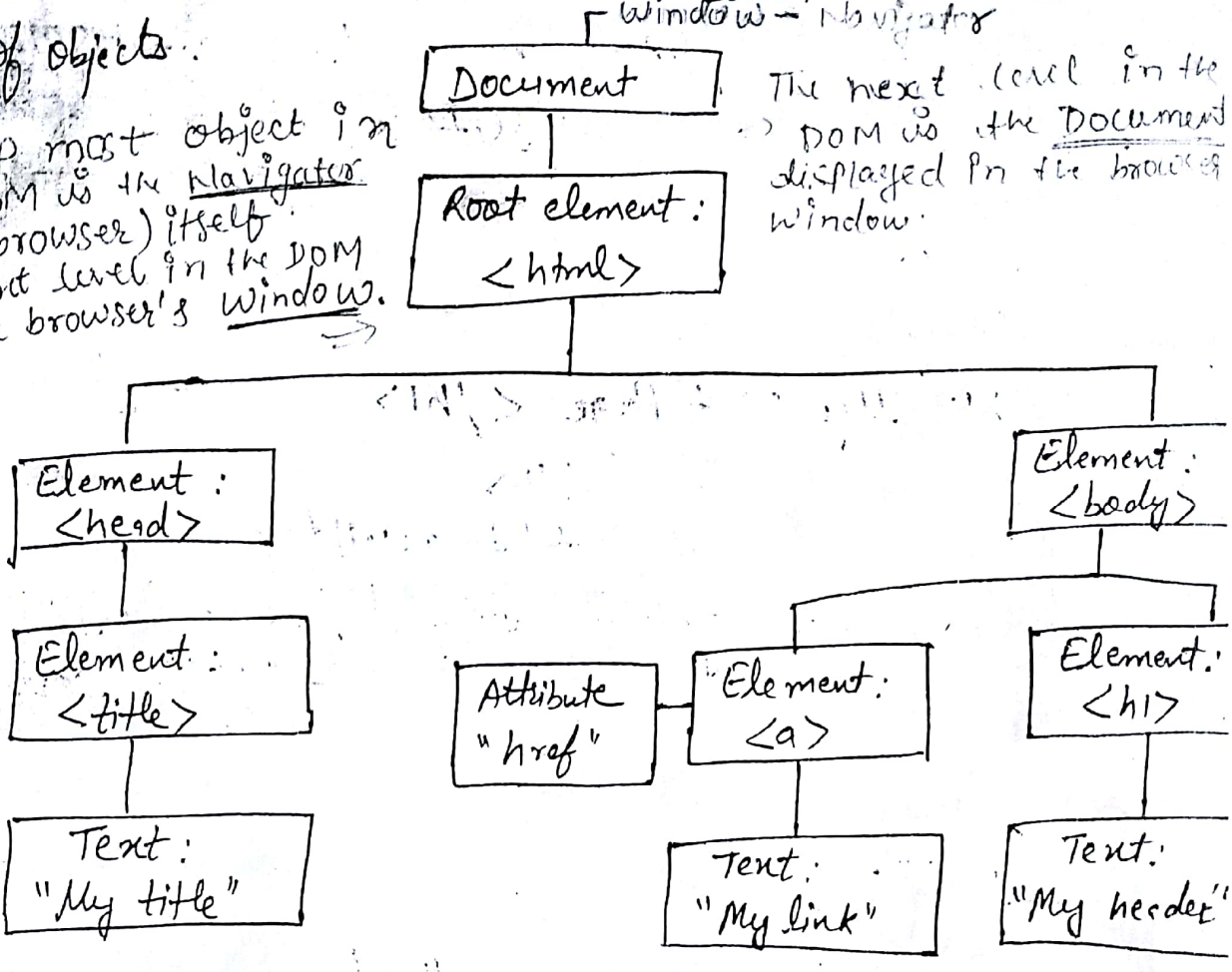
The HTML DOM (Document Object Model) :-

→ The HTML DOM model is constructed as a tree of objects.



The top most object in the DOM is the Navigator (i.e. browser) itself.
The next level in the DOM is the browser's window.

The next level in the DOM is the Document displayed in the browser window.

→ DOM is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure and style of a document.

→ The HTML DOM is a standard object model and programming interface for HTML. It defines:
- The HTML elements as objects
- The properties of all HTML elements
- The methods to access all HTML elements
- The events for all HTML elements.

⇒ The HTML DOM is a standard for how to get, change, add or delete HTML elements.

⇒ HTML DOM methods are actions we can perform on HTML elements.

⇒ HTML DOM properties are values of HTML elements that we can set or change.

Ex
```
<html>
<body>
<h1> My First Page </h1>
<p id = "demo"> </p>
<script> type = "text/javascript">
    document.getElementById("demo").innerHTML
                = "Hello world!";
</script>
</body>
</html>
```

⇒ The getElementById method :- used id of the element to find the element.

→ Used for accessing an HTML element with the help of its id.

⇒ The innerHTML property :- Used to get the content of an element.

→ The innerHTML property is useful for getting or replacing the content of HTML elements.

→ The innerHTML property can be used to get change any HTML element, including <html> and <body>.

DOM document :- It is the owner of all other objects in our web page.

→ If we want to access any element in an HTML page, we always start with accessing the document object.

Finding HTML Elements

Method	Description
✓ document.getElementById(id)	Find an element by element
document.getElementsByTagName(name)	Find elements by tag name
document.getElementsByClassName(name)	Find elements by class nam
✓ document.write(text)	Write into the HTML output
document.createElement(element)	Create an HTML element
✓ element.innerHTML = new html content	Change the inner HTML of an elem
element.attribute = new value	Change the attribute value of an HTML elements.

var btn = document.createElement("BUTTON");
document.body.appendChild(btn);
var t = document.createTextNode("Click");
btn.appendChild(t);

Ex
```
<html>
<body>
<p id = "intro"> Hello world! </p>
<p> The example demostrate the <b> getElementById </b>
    method !!! </p>
<p id = "demo"> </p>
<script type = "text/javascript">
    var myElement = document.getElementById("intro");
    document.getElementById("demo").innerHTML =
        myElement.innerHTML + "This is the new text
        we want to append";
</script>
</body>
```

⇒ If the element is found, the method will return the element as an object (in myElement).
⇒ If the element is not found, myElement will contain null.

⇒ Changing HTML Content :-

→ The easiest way to modify the content of an HTML element is by using the innerHTML property.

Syntax

```
document.getElementById(id).innerHTML = new HTML
```

Example

```
<html>
<body>
<h1 id = "header"> Old Header </h1>
<script type = "text/javascript">
{var element = document.getElementById("head
element.innerHTML = "New Header";
document.getElementById("header").innerHTML = "New H
</script>
</body>
</html>
```

⇒ changing the value of an Attribute
⇒ onclick event on Button.

Ex

```
<!DOCTYPE html>
<html>
<body>
<img id="image" src="sunset.jpg" width="160" height="120">
<script type="text/javascript">
function myfunc()
{ document.getElementById("image").src
                "winter.jpg";
}
</script>
<p> The original image was sunset.jpg
but the script changed it to winter.
when click on the button" </p>
<button type="button" name="b1"
        onclick="myfun()"> click here to
change image </button>
</body>
</html>
```

⇒ changes the style of the HTML element with id = "id1", when the user clicks a bu

```
<!DOCTYPE html>
<html>
<body>
<h1 id = "id1"> My heading 1 </h1>
<button type = "button" onclick = "document
    .getElementById("id1").style.color = 'red'">
    Click me! </button>
</body>
</html>
```

JavaScript Events :-

→ HTML events are "things" that happen to HTML elements.

→ JavaScript can react on these events.

→ Examples of HTML events :-
- when a user clicks the mouse
- when a web page has loaded
- when an image has been loaded
- when the mouse moves over an element
- when an input field is changed
- when an HTML form is submitted
- when a user strokes a key.

→ List of some common HTML events :-

Event	Description
onclick	The user clicks an HTML element
onchange	An HTML element has been changed
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onmousedown	The user pushes a keyboard key / Press mouse but
onmouseup	The user release the mouse but
onload	The browser has finished loading page.
onsubmit	Try to submit a form
onblur	Triggers when element loses focus
onfocus	Triggers when element gets focus

Onfocus :- when an input field gets focus

```
<html>
<head>
<script type = "text/javascript">
  function myfunction(x)
  { x.style.background = "yellow";
  }
</script>
</head>
<body>
  Enter your name :
  <input type = "text" onfocus = "myfunction(this)">
  <p> when the input field gets focus
  a function is triggered which changes
  the background-color. </p>
</body>
</html>
```

Onblur :- When a uses leaves an input field

```
<script type = "text/javascript">
  function myfunction()
  { var x = document.getElementById("fname")
    x.value = x.value.toUpperCase();
  }
</script>
<body>
  Enter your name :
  <input type = "text" id = "fname" onblur = "myF
  <p> When we leave the input field, a func
  is triggered which transforms the input text
  </p>
```

Onchange :- when a user selects a dropdown values.

```
<html>
<head>
<script type = "text/javascript">
function preferedBrowser()
{ prefer = document.forms[0].browsers.val
  alert("we prefer browsing internet with"
        prefer);
}
</script>
</head>
<body>
<form>
Choose which browser you prefer :
<select id = "browsers" onchange = "preferedBr
<option value = "Chrome"> chrome </option>
<option value = "Internet Explorer"> Interne
  Explorer </option>
<option value = "Firefox"> Firefox. </option>
</select>
</form>
</body>
</html>
```

Onclick :- change text on click on the text

```
<html>
<body>
<h1 onclick = "this.innerHTML = 'JavaScript'">
      click on this text </h1>
</body>
```

onmouseover :- The event occurs when the pointer is moved onto an element.

onmouseout :- The event occurs when the pointer moved out of an element.

```
<html>
<body>
<div onmouseover = "mOver(this)"
     onmouseout = "mOut(this)"
     style = "background-color : #D94A38;
              width : 120px; height : 20px;
              padding : 40px;">
Mouse Over Me </div>
<script type = "text/javascript">
function mOver(obj)
{
  obj.innerHTML = "Thank you"
}
function mOut(obj)
{
  obj.innerHTML = "Mouse Over Me"
}
</script>
</body>
</html>
```

onmousedown :- The event occurs when the user presses a mouse button over an element.

onmouseup :- The event occurs when a user releases a mouse button over an element.

```
<html>
<body>
<div onmousedown = "mDown(this)"
onmouseup = "mUp(this)"
style = "background-color : #D94A38 ;
width : 90px ; height : 20px ;
padding : 40px ;"> click me </div>

<script>
function mDown(obj)
{ obj.style.backgroundColor = "#_lec5e5"
  obj.innerHTML = "Release Me";
}
function mUp(obj)
{ obj.style.backgroundColor = "#D94A38";
  obj.innerHTML = "Thank you";
}
</script>
</body>
</html>
```

⇒ JavaScript Form Validation :-

It is important to validate the form submitted by the user because it can have inappropriate values. So validation is must.

→ The JavaScript provides us the facility the validate the form on the client side so processing will be fast than server-side validation. So, most of the web developers prefer JavaScript form validation.

→ In JavaScript, we can validate name, password, email, date, mobile nb. etc. fields.

⇒ Javascript form validation ~~Example~~ :-

→ We are going to validate the name & password. The name can't be empty and password can't be less than 6 char. long.

→ Here, we are validating the form on form submit. The user will not be forwarded to the next page until given values are correct.

```
<html>
<body>
<script type = "text/javascript">
    function validateform()
    {
        var name = document.myform.name.value
        var password = document.myform.
                        password.value;

        if (name == "")
        {   alert ("Name can't be blank");
            return false;
```

```
else if (password.length < 6)
{ alert ("Password must be at least 6 char.");
    return false;
}
}
</script>
<body>
<form name = "myform" method = "post"
action = "http://www.javascriptpages.jsp"
onsubmit = "return validateform()">
Name : <input type = "text" name = "name"><br/>
Password : <input type = "password" name = "password"><br/>
<input type = "submit" value = "register">
</form>
</body>
</html>
```

⇒ JavaScript Retype Password validation

```
<html>
<head>
<script type = "text/javascript">
function matchpass()
{
var firstpassword = document.f1.password.value;
var secondpassword = document.f1.password2.value;
```

```
if (firstpassword == secondpassword)
{   return true;

}
else
{   alert ("password must be same!");
    return false;
 }
}
</script>
</head>
<body>
<form name = "f1"
 action = "http://pages.jsp"
 onsubmit = "return matchpass()">
 Password : <input type = "password"
                   name = "password"/> <br>

 Re-enter Password : <input type = "password"
                   name = "password2"/> <br/>

 <input type = "submit">

</form>
</body>
</html>
```

⇒ JavaScript Number Validation :-

Let's validate the textfield for numeric value. Here, we are using isNaN() function.

```
<html>
<head>
<script type = "text/javascript">
  function validate()
  { var num = document.myform.num.value
    if (isNaN(num))
    { document.getElementById("numloc").innerHTML
        = "Enter Numeric value only";
      return false;
    }
    else { return true;
    }
  }
</script>
</head>
<body>
<form name = "myform" action = "http://page.jsp"
  onsubmit = "return validate()">
Number: <input type = "text" name="num">
  <span id = "numloc"></span> <br>
<input type = "submit" value = "submit">
</form>
</body>
</html>
```

first converts the tested value to a number, then tests it.

⇒ **JavaScript email validation :-**

We can validate the email by the help of JavaScript.

There are many criteria that need to be follow to validate the email id such as:

1. email id must contain the @ and . character.
2. There must be at least one character before and after the @.
3. There must be at least two characte after . (dot).
4. .

```
if (atposition < 1 || dotposition < atposition +
        || dotposition + 2 >= x.length )
{  alert ("Please enter a valid email address");
}
```

→

```
<html>
<body>
<script type = "text/ javascript">
function validateemail()
{
    var x = document.myform.email.value;
    var atposition = x.indexOf ("@");
    var dotposition = x.lastIndexOf (".");
```

```
if (atposition == -1) || (dotposition == -1) ||
    (atposition > dotposition)
{ alert ("Please enter a valid e-mail address");
  return false;
}
}
</script>
<body>
<form name = "myform" method = "post"
action = "http://www.javascriptpage.jsp"
onsubmit = "return validatemail();">
Email: <input type = "text" name = "email">
<br/>
<input type = "submit" value = "register">
</form>
</body>
</html>
```

(9)

invalid email;
(1) @index < 1 (2) dot index >= (str.length - 2
(3) dotindex - atindex < 3 ⇒ atleast 2 characters between there atleast 2 characters
if -ve i.e. dot is behind @

at least 2 characters after

```
<script>
function validate()
{ var result = true;
var e = document.getElementsByName("email")[0].value;
var atindex = e.indexOf('@');
var dotindex = e.lastIndexOf('.');
if (atindex < 1 || dotindex >= e.length - 2 || dotindex - atindex < 3)
    result = false
return result;
</script> <body> <form onsubmit = "return validate()">
Email id: <input type = "text" name = "email" required>
<input type = "submit" value = "register" />
</body> </form>
```

Window Object :-

→ Javascript lets us create and manipulate windows for presenting HTML content.

→ The window object is the top-level object in the Javascript client hierarchy.

⇒ Open and close a new window and checked whether the window is opened or closed

⇒ Opening and closing windows :-

```
<html>
<head>
<title> Javascript window object </title>
</head>
<body> type = "text/javascript">
<script
function newwindow()
{ win1 = window.open(); win1 = window.open("s.html");
//win1.document.write("<p> we are learning to
        open a new window </p>");
}
function Closewindow()
{ win1.close();
}
function testwindow()
{ if (win1.closed)
  { alert ('window is closed...');
  }
  else
  { alert ('window is open...');
  }
```

"http://www.google.com"

```
<form name = "form1">
<input type = "button" name = "bt1" value = "open new
    window." onclick = 'newwindow()' />
<br/>
<input type = "button" name = "bt2" value = "close
    the window." onclick = 'closewindow()' />
<br/>
<input type = "button" name = "bt3" value = "Test:
    window is open or not?" onclick = 'testwindow
<br/>
</form>
</body>
</html>
```

⇒ Open a new window and resize the width and height to 250 px. — move the new window 250px relative to its current

```
<html>
<body>
<button onclick = "openWin()" > Create window </button>
<button onclick = "moveWin()" > Move myWindow </button>
<button onclick = "resizeWin()" > Resize window
</button>
<script type = "text/javascript">
var myWindow;
```

```
function openWin()
{  myWindow = window.open("", "", "width=200,
                          height=200");
}
function resizeWin()
{   myWindow.resizeBy(250, 250);
    myWindow.focus();
}
</script>
</body>
</html>
function moveWin()
{   myWindow.moveBy(250, 250);   myWindow.move
    myWindow.focus();                (500, 100);
}
```

⇒ program to display the window's height & width

```
<html>
<body>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script type="text/javascript">
function myFunction()
{   var w = window.innerWidth;
    var h = window.innerHeight;
    document.getElementById("demo").innerHTML
        = "Width: " + w + "<br>Height: " + h;
}
</script>
</html>
</body>
```

Wap to click the button to create a window, display the name of the new window and write some text in the new windows & the source (parent) window.

```
<html>
<body>
<button onclick = "myFunction1()"> Try it
</button>
<button onclick = "myFunction2()"> click here
</button>
<script type = "javascript">
function myFunction1()
{ var myWindow = window.open("", "MsgWindo
    "width = 200, height = 100");
  myWindow.document.write("<p> This window
    name is: " + myWindow.name + "</p>");
}
function myFunction2()
{ var myWindow = window.open("", "myWindow",
    "width = 200, height = 100");
  myWindow.document.write("<p> This is
    'myWindow' </p>");
  myWindow.opener.document.write("<p>
    This is the source window </p>");
}
</script> </body> </html>
```

Scanned by CamScanner

Some window object Properties :-

① **closed** :- Returns a Boolean value indicating whether a window has been closed or not.

② **document** :- Returns the Document object for the window.

③ **innerHeight** :- Returns the inner height of a window's content area.

④ **innerWidth** :- Returns the inner width of a windows's content area.

⑤ **name** - sets or returns the name of a window

⑥ **opener** :- Returns ~~the~~ a reference to the window that created the window.

⑦ **parent** :- Returns the parent window of the current window.

⑧ **self** :- Returns the current window.

⑨ **top** :- Returns the topmost browser window.

Window Object Methods :-

1. alert() :- Displays an alert box with a message and an OK button.
2. blur() :- Removes focus from the current window
3. close() :- closes the current window.
4. confirm() :- Displays a dialog box with a message and an OK & a Cancel button
5. focus :- Sets focus to the current window.
6. moveBy() :- Moves a window relative to its current position.
7. moveTo() :- Moves a window to the specified position.
8. open() :- Opens a new browser window
9. print() :- Prints the content of the current window
10. prompt() :- Displays a ~~dialog~~ dialog box that prompts the visitor for input
11. resizeBy() :- Resizes the window by the specified pixels relative to its current position
12. resizeTo() :- Resizes the window to the specified width & height.

⇒ Navigator object :-

→ The client-side JavaScript objects are referred to as Navigator objects, to distinguish them from server-side objects or user-defined objects.

→ The window.navigator object contains information about the visitor's browser.

→ The window.navigator object can be written without the window prefix.

Properties

~~(1) cookieEnabled~~

(1) appName :- The appName property returns the application name of the browser.

Ex
```
<html>
<body>
<h1> The Navigator object </h1>
<p id = "demo" ></p>
<p> Netscape is the application name for
    Chrome, Firefox and Safari. </p>
<script>
    document.getElementById("demo").innerHTML
        = "navigator.appName is " + navigator.appName;
</script>
</body>
</html>
```

appVersion :- The appVersion property returns version info. about the browser.

```
<html>
<body>
<h1> The navigator object </h1>
<p id = "demo" ></p>
<script>
    document.getElementById("demo").innerHTML
        = navigator.appVersion;

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

(3) **appCodeName** :- This property returns the application code name of the browser.

```
<script>
    document.getElementById("demo").innerHTML
        = "navigator.appCodeName is " +
          navigator.appCodeName;

</script>
```

(4) **userAgent** :- The userAgent property returns the user-agent header sent by the browser to the server.

```
<script>
    document.getElementById("demo").innerHTML =
        navigator.userAgent;

</script>
```

(5) **cookieEnabled** :- This property returns true if cookies are enabled, otherwise false:

```
<script>
    document.getElementById("demo").innerHTML
    = "cookiesEnabled is " + navigator.cookieEnabled
</script>
```

(6) **product** :- The product property returns the product name of the browser engine:

```
<script>
    document.getElementById("demo").innerHTML
    = "navigator.product is " + navigator.product;
</script>
```

(7) **platform** :- The platform property returns the browser platform (operating system):

```
<script>
    document.getElementById("demo").innerHTML =
    navigator.platform;
</script>
```

(8) **language** - The language property returns the browser's language.

```
<script>
    document.getElementById("demo").innerHTML =
    navigator.language;
</script>
```

online :- The online property returns true if the browser is online. (13)

```
<script>
    document.getElementById("demo").innerHTML =
        navigator.online;
</script>
```

Method :-

① **javaEnabled()** :- This method returns true if Java is enabled.

```
<script>
    document.getElementById("demo").innerHTML
        = navigator.javaEnabled();
</script>
```

⇒ The History Object :-

→ The history object contains the URLs visited by the user (within a browser window).

→ The history object is part of the window object and is accessed through the window.history.property.

Property

Length :- Returns the number of URLs in the history list of the current browser window.

→ The property returns at least 1, because the list includes the current loaded page.

→ This property is useful to find out how many pages the user has visited in the current browsing session.

→ Max. length is 50. This property is read-only

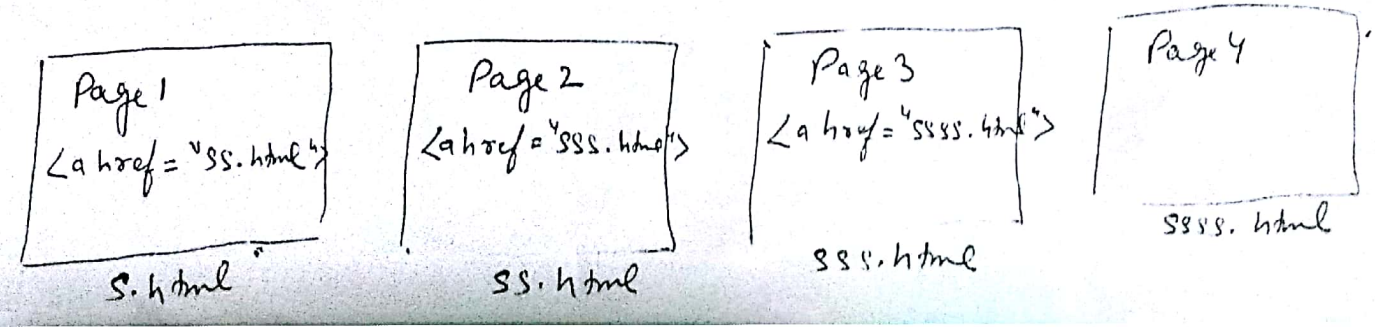
Methods

back() :- Loads the previous URL in the history list

forward() :- Loads the next URL in the history list

go() :- Loads a specific URL from the history list.

```
</html>
<body>
<h1 style = "color : red;"> Page 1 </h1> <br/>
<button onclick = "goBackward ()"> Go Backward
</button>
<button onclick = "goForward ()"> Go Forward
</button>
<button onclick = "goBack ()"> Go 2 pages back
</button> <br/> <br/>
<a href = "ss.html"> click for new page </a>
<script type = "text/javascript">
function goForward ()
{  window. history. forward();
}
function goBackward ()
{  window. history. back ();
}
function goBack ()
{  window. history. go (-2);
}
</script>
</body>
</html>
```



⇒ The HTTP is designed to enable communi-cation between clients and servers.

→ HTTP works as a request-response protocol between a client and server.

→ Two HTTP Request Methods are :-

GET and POST

⇒ GET - Requests data from a specified resource

⇒ POST - Submits data to be processed to a specified resource.

⇒ The GET Method

The query string (name/value pairs) is sent in the URL of a GET request.

⇒ The POST method

The query string (name/value pairs) is sent in the HTTP message body of a POST request.

* Forms access. 3 methods :-

① document.formname.

② doc. form(1) (0)

③ do form[formname].