OBJECT™
TECHNOLOGIES

DOM Manipulation

What will be covered

OBJECT

- Modifying HTML Content
- Modifying attributes
- Modifying style
- Adding New HTML Elements
- Removing Existing HTML Elements
- Replacing existing element
- Browser objects
- Window object
- Navigator object
- History object
- Screen object
- Location object

Modifying HTML Content

OBJECT

- The easiest way to modify the content of an HTML element is by using the **innerHTML** property.
- To change the content of an HTML element, use this syntax:

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

```
<body>

<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML = "New text!";
</script>

</body>
```

Modifying attributes

OBJECT TECHNOLOGIES

- To change the value of an HTML attribute, use this syntax:

  document.getElementById(*id*).*attribute* = *new value*

  The following code changes src of the image to some different image.

```
<body>

<img id="myImage" src="smiley.gif">

<script>
document.getElementById("myImage").src = "landscape.jpg";
</script>

</body>
```

Modifying Style

OBJECT

- To change the style of an HTML element, use this syntax:

  document.getElementById(*id*).style.*property* = *new style*

```
<p id="p2">Hello World!</p>

<script>
document.getElementById("p2").style.color = "blue";
</script>

<p>The paragraph above was changed by a script.</p>
```

- *Note:*

  CSS attributes which are hyphen separated should be written like this:

  background-color should be written as backgroundColor

  margin-left should be written as marginLeft

Adding New HTML Elements (Nodes)

OBJECT

- To add a new element to the HTML DOM, you must create the element (element node) first, and then append it to an existing element.

```
<div id="div1">
    <p id="p1">This is a paragraph.</p>
    <p id="p2">This is another paragraph.</p>
</div>

<script>
    var para = document.createElement("p");    //creates a new node with p element
    var node = document.createTextNode("This is new.");   //creates a text node for paragraph
    para.appendChild(node);

    var element = document.getElementById("div1");
    element.appendChild(para);    //appending paragraph in division
</script>
```

Removing Existing HTML Elements

OBJECT

To remove an HTML element, you must know the parent of the element

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
var parent = document.getElementById("div1");
var child = document.getElementById("p1");
parent.removeChild(child);
</script>
```

Replacing existing element

OBJECT TECHNOLOGIES

- The replaceChild() method of the Node element replaces a child node within the given (parent) node.
- Syntax : replaceChild(newChild, oldChild); where newChild is the new node to replace oldChild and oldChild is The child to be replaced.
- If the new node is already present somewhere else in the DOM, it is first removed from that position.

```
// select the element that will be replaced
var el = document.querySelector('div');

// <a href="/javascript/manipulation/creating-
var newEl = document.createElement('p');
newEl.innerHTML = '<b>Hello World!</b>';

// replace el with newEL
el.parentNode.replaceChild(newEl, el);
```

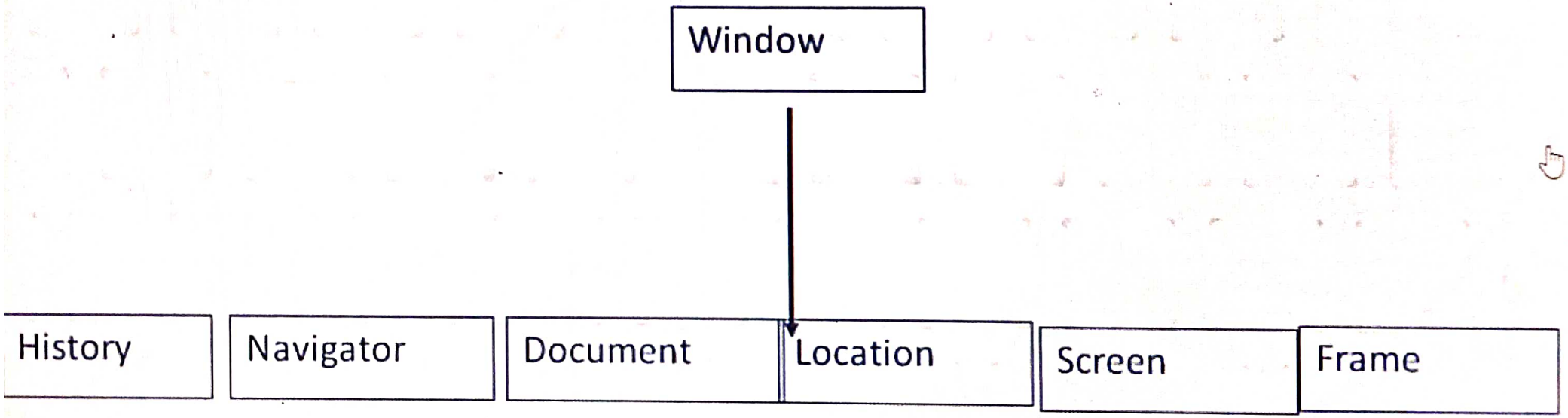
Browser objects

OBJECT

- 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.
- The Iterfaces (object types) which help us interact with the browser window are known as Browser objects.
- The Browser objects are automatically created by a browser at the time of loading a Web page.
- Following are the broswer 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

- The window object that serves as the global object and global execution context for client side JavaScript code.
- The window object is the top-level object for each window or frame and is the parent object for the document, location, and history objects.



Window object

OBJECT TECHNOLOGIES

Window properties :

- defaultStatus
- length
- name
- opener
- parent
- self
- status
- top
- window

Window object

Window methods :

- alert(message)
- blur()
- close()
- focus()
- navigator(url)
- open(url,name,features)
- prompt(message,response)
- setTimeout(expression,time)
- scroll(x,y)
- clearTimeout(name)

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.

Properties : appcodeName

appName

appVersion

Methods : javaEnabled()

History Object

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.
- The history object allows a script to work with the Navigator browser's history list in JavaScript. For security and privacy reasons, the actual content of the list is not reflected into JavaScript.
  - Properties
    - length
  - Methods :
    - back()
    - forward()
    - go(*location*)

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 using **window.screen**
- It provides information about the dimensions of the display screen such as its height, width, color bits, etc.
- Available properties are :
  - availHeight - excluding window's task bar
  - availWidth - excluding window's task bar
  - height - total height
  - pixelDepth - color Resolution, in bits per pixel,
  - width - total width

Location object

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.
- It reflects information about the current URL
  - Properties
    - hash
    - host
    - hostname
    - href
    - pathname
    - port
    - protocol
    - search
  - Methods : reload()
    replace(*url*)

DOM Manipulation

DOM manipulation in javascript is an important factor while creating a web application using HTML and JavaScript. It is the process of interacting with the DOM API to change or modify an HTML document that will be displayed in a web browser. This HTML document can be changed to add or remove elements, update existing elements, rearrange existing elements, etc.

Modifying Content

Many HTML elements have the ability to display some text. Examples of such elements are our headings, paragraphs, sections, inputs, buttons, and many more. The text content of these elements can be modified by following ways :

1. By using innerText property

```
var myheading = document.getElementById("main_head");
myheading.innerText = "Modified header";
```

innerText property of a node returns string which can be even modified by assigning some string to it.

2. By using textContent property

The textContent property of the Node interface represents the text content of the node and its descendants.

Value of textContent property will be a string or null. If the node is a document or a doctype, textContent returns null. For other node types, textContent returns the concatenation of the textContent of every child node.

textContent gets the content of all elements, including <script> and <style> elements. In contrast, innerText only shows "human-readable" elements.

Element.innerHTML returns HTML, as its name indicates. Sometimes people use innerHTML to retrieve or write text inside an element, but textContent has better performance because its value is not parsed as HTML.

Suppose that you have the following HTML snippet:

```
<div id="note">
   JavaScript textContent Demo!
   <span style="display:none">Hidden Text!</span>
   <!-- my comment -->
</div>
```

The following example uses the textContent property to get the text of the element:

```
let note = document.getElementById('note');
console.log(note.textContent);
```

```
<div id="note">
  JavaScript textContent Demo!
  <span style="display:none">Hidden Text!</span>
  <!-- my comment -->
</div>
```

The following example uses the textContent property to get the text of the element:

```
let note = document.getElementById('note');
console.log(note.textContent);
```

As you can see clearly from the output, the textContent property returns the concatenation of the textContent of every child node,

Besides reading textContent, you can also use the textContent property to set the text for a node:

```
node.textContent = newText;
```

3. By using innerHTML property

The innerHTML is a property of the Element that allows you to get or set the HTML markup contained within the element:

To get the HTML markup contained within an element, you use the following syntax:

```
let content = element.innerHTML;
```

Suppose that you have the following markup:

```
<ul id="menu">
  <li>Home</li>
  <li>Services</li>
</ul>
```

The following example uses the innerHTML property to get the content of the <ul> element:

```
let menu = document.getElementById('menu');
console.log(menu.innerHTML);
```

To set the value of innerHTML property, you use this syntax:

```
element.innerHTML = newHTML;
```

The setting will replace the existing content of an element with the new content.

4. By using firstChild.nodeValue property

The nodeValue property sets or returns the node value of the specified node.

If the node is an element node, the nodeValue property will return null.

If you want to return the text of an element, remember that text is always inside a Text node, and you will have to return the Text node's node value

Return the node value:

```
node.nodeValue
```

Set the node value:

```
node.nodeValue=value
```

Example:

```
var parr = document.getElementsByTagName("p");
for (i = 0; i < parr.length; i++) {
        //equivalent with innerHTML
        parr[i].firstChild.nodeValue = "This is paragraph no " + (i + 1);
        //parr[i].innerHTML = "This is paragraph no " + (i + 1);
}
```

Modifying Style

Changing an element's style (CSS) attributes is one of the ways in which the DOM can be used. For example, in the code below, the onfocus event handler method is used to change the <input> element's style when the user clicks inside it. When this event occurs, the element's background color will change to purple.

```
<!DOCTYPE html>
<html>
<body>
   Email: <input type="text" id="outline" onfocus="background()">
   <p>Click in the text input to focus the element.<br/>
   This will trigger the 'background' function, which changes the element's background color to purple.</p>
   <script>
   function background() {
```

Modifying Style

Changing an element's style (CSS) attributes is one of the ways in which the DOM can be used. For example, in the code below, the onfocus event handler method is used to change the <input> element's style when the user clicks inside it. When this event occurs, the element's background color will change to purple.

```
<!DOCTYPE html>
<html>
<body>
  Email: <input type="text" id="outline" onfocus="background()">
  <p>Click in the text input to focus the element.<br/>
  This will trigger the 'background' function, which changes the element's background color to purple.</p>
  <script>
  function background() {
    document.getElementById("outline").style.background = "purple";
  }
</script>
</body>
</html>
```

The general syntax is:

```
element.style.property= value
```

HTML elements can be modified with respect to their attributes :

Consider following page design :

```
<body>
  <img src="https://js-tutorials.com/shark.png">
</body>
const img = document.querySelector('img');
img.hasAttribute('src');          // returns true
img.getAttribute('src');          // returns "...shark.png"
img.removeAttribute('src');       // remove the src attribute and value
```

This image can be assigned with new value with img.setAttribute():

```
img.setAttribute('src', 'https://js-tutorials.com/octopus.png')
```

HTML elements can be modified with respect to their attributes :

Consider following page design :

```
<body>
   <img src="https://js-tutorials.com/shark.png">
</body>
const img = document.querySelector('img');
img.hasAttribute('src');          // returns true
img.getAttribute('src');          // returns "...shark.png"
img.removeAttribute('src');       // remove the src attribute and value
```

This image can be assigned with new value with img.setAttribute():

```
img.setAttribute('src', 'https://js-tutorials.com/octopus.png')
```

CSS classes are used to apply styles to multiple elements, unlike IDs which can only exist once per page. In JavaScript, we have the className and classList properties to work with the class attribute.

Styles can be modified by changing the class name associated with the element.

Property	Purpose
element.className	Gets or sets class value
element.classList.add('active')	Adds one or more class values
element.classList.toggle('active')	Toggles a class on or off
element.classList.contains('active')	Checks if class value exists
element.classList.replace('old', 'new')	Replace an existing class value with a new class value
element.classList.remove('active')	Remove a class value

Example :

Consider following design :

```
<style>
```

Example :

Consider following design :

```
<style>
  body {
    max-width: 600px;
    margin: 0 auto;
    font-family: sans-serif;
  }
  .active {
    border: 2px solid blue;
  }
  .warning {
    border: 2px solid red;
  }
</style>
<body>
  <div>Div 1</div>
  <div class="active">Div 2</div>
</body>
// Select the first div
const div = document.querySelector('div');
// Assign the warning class to the first div
div.className = 'warning';
```

Handling Browser Objects

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

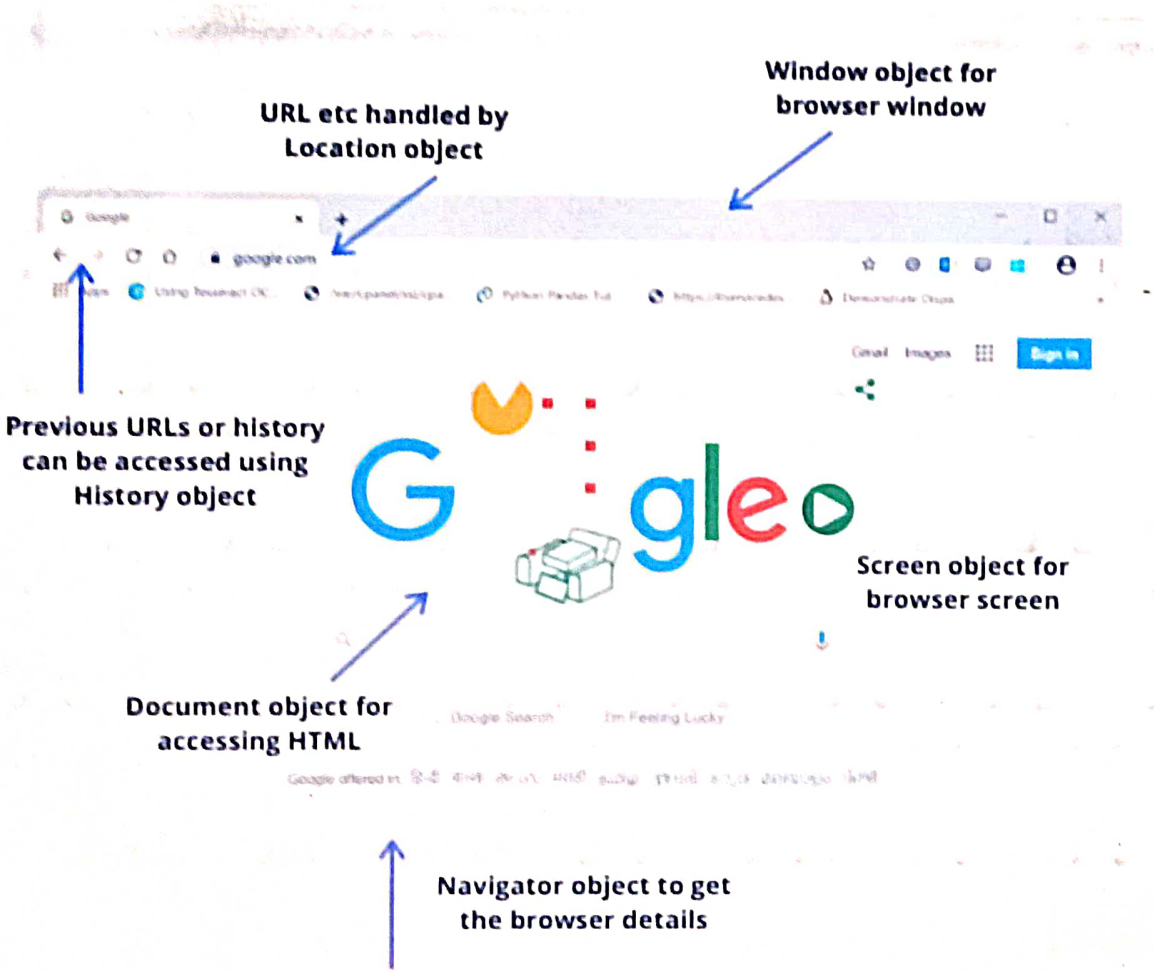
For example, when an HTML document is opened in a browser window, the browser interprets the document as a collection of hierarchical objects(HTML tags) and accordingly displays the data contained in these objects(HTML page rendering). The browser parses the document and creates a collection of objects, which defines the documents and its details. Following are various objects that are available which allows to access different parts of browser :

Window object for

Handling Browser Objects

The Interfaces (object types) which help us interact with the browser window are known as Browser objects. Browser object is not an official term but its 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. Following are various objects that are available which allows to access different parts of browser :



Browser Objects:

The objects listed below are called browser objects.

Browser Objects:

The objects listed below are called browser objects.

1. Window - part of DOM API
2. Navigator
3. Document - part of DOM API
4. Screen - property of Window object
5. History - property of Window object
6. 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.

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.

```
//client's browser and its settings
alert(navigator.appCodeName)
alert(navigator.appVersion)
alert(navigator.platform)
alert(navigator.cookieEnabled)
```

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.

```
alert(history.length);
history.go(-1);
history.back();
```

Screen Object

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.

```
//screen - info about monitor
alert(screen.availWidth);
alert(screen.availHeight);
```

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.

Assignments

1. Display the following information in the form of Unordered List. And on the click event of the button replace the list content with their corresponding image icons.

   Desktop

   MyComputer

   Folder

   Browser

   WordDocument

   (Hint : Use replaceChild() for replacing li with img)

2. Convert the following table such that each row should be displayed as a separate division containing heading, image and bold text.

Cricket		Cricket Cricket Cricket Cricket Cricket Cricket Cricket
Football		Football Football Football Football Football Football Football
		Volleyball Volleyball

Assignments

1. Display the following information in the form of Unordered List. And on the click event of the button replace the list content with their corresponding image icons.

   Desktop

   MyComputer

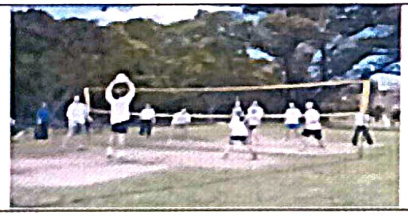
   Folder

   Browser

   WordDocument

   (Hint : Use replaceChild() for replacing li with img)

2. Convert the following table such that each row should be displayed as a separate division containing heading, image and bold text.

Cricket		Cricket Cricket Cricket Cricket Cricket Cricket Cricket
Football		Football Football Football Football Football Football Football
Volleyball		Volleyball Volleyball Volleyball Volleyball Volleyball Volleyball Volleyball

3. Detect browser used by client and its version
4. Check whether cookies are enabled or disabled on the client"s browser
5. Design a web page as follows :

   Add a button which contains text as "Show Window". When user clicks on this button then open a new window and change the text of button as "Hide Window".

   Accept the height and width from user and accordingly change the size of window.

   Also accept the x and y coordinate position and accordingly move the window to that new location.

   Accept some text from user and write that text in that new window.

Cricket		Cricket Cricket Cricket Cricket Cricket Cricket Cricket
Football		Football Football Football Football Football Football Football
Volleyball		Volleyball Volleyball Volleyball Volleyball Volleyball Volleyball Volleyball

3. Detect browser used by client and its version
4. Check whether cookies are enabled or disabled on the client"s browser
5. Design a web page as follows :

   Add a button which contains text as "Show Window". When user clicks on this button then open a new window and change the text of button as "Hide Window".

   Accept the height and width from user and accordingly change the size of window.

   Also accept the x and y coordinate position and accordingly move the window to that new location.

   Accept some text from user and write that text in that new window.

   Create a button to close new opened window and get the confirmation from user before closing