**DOM (Document Object Model)**

**What scripting interfaces are available?**

The most basic scripting interface developed at **W3C** is the DOM, the **D**ocument **O**bject **M**odel which allows programs and scripts to dynamically access and updates the content, structure and style of documents. DOM specifications form the core of DHTML.

API => lib => collection of predefined programs (classes)

Modifications of the content using the DOM by the user and by scripts trigger events that developers can make use of to build rich user interfaces.

> DOM is the tree structure of html elements(tags) that are present within the web page.

> When the web page has opened/loaded in browser, DOM will be automatically created by the browser.

>the changes made to DOM are called as "DOM manipulations". DOM manipulations are performed by using JavaScript.

>DOM is a platform and language neutral interface that allows the programs/script to dynamically access & updates the content structure and style of the document.

> The entire browser is called as "window", the webpage running on the browser is called as "document". it has only one main element called "html", "html" has two children "head" and "body". there are many children for both "head" and "body".

Lib=> classes

Html=> tags

Dom has the following detailed structure

No.of html tags = no.of DOMs + extra tag

Tag Name DOM Class

head 🡺 HTMLHeadElement 🡺object

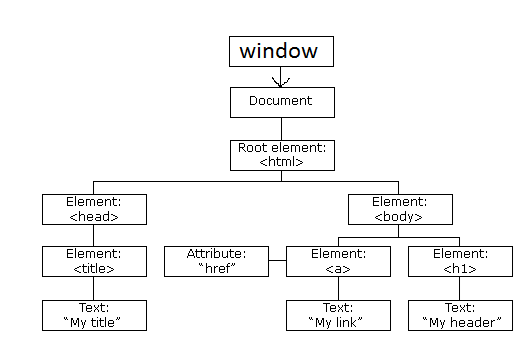
body 🡺 HTMLBodyElement 🡺 object

p 🡺 HTMLParagraphElement

input 🡺 HTMLInputElement

h1 to h6 🡺 HTMLHeadingElement

button 🡺 HTMLButtonElement





The above structure represents as follows:

1. The entire document is a document node
2. Entry HTML element/tag is an element node
3. The text inside HTML elements are text node
4. Every HTML attribute is an attribute node

**Objects in DOM:**

* **document (**this object providing interaction with webpage/web document**)**
* **element (**this object providing interaction with tags/elements**)**

**every object it contains properties and methods.**

**object.property**

**object.method()**

**document object**

**>this object providing interaction with current webpage/webdocument.**

**> this "document" has some properties and methods**

1. **title**

this property represents title of the current working/viewing web page in browser window.

Syn: **document.title**

**document.title=”New title”;**

1. **head**

this property represents the "head" section of current working/viewing web page in browser window.

Syn: refvar= **document.head**

1. **body**

this property represents the "body" section of current working/viewing web page in browser window.

Syn: **document.body**

1. **images**

this property represents all "images" of current web page, as an array format.

Syn: **document.images**

**10imgs 🡺HTMLImageElement class 🡺 10 objs =>init => assign to collection obj**

1. **links**

this property represents all hyperlinks (<a> tags) of current web page, as an array format.

Syn: **document.links**

1. **url**

this property represents url of current web page.

Syn: **document.URL**

1. **write()**

this method displays message in the web page.

Syn: **document.write(...)**

1. **writeln()**

this method displays message in the web page.

Syn: **document.writeln(...)**

1. **getElementById()**

this methodreturns a reference(object) of element/tags of a specified id.

by using that reference we can access properties of that element and we call methods.

Syn: **document.getElementById("id")**

1. **getElementsByName()**

this methodreturns array of elements/tags which have same name (attribute).

this spe used in checkbox and radio button case.

Syn: **document.getElementsByName("name")**

1. **getElementsByTagName()**

this methodreturns array of elements/tags which have same tag name.

Syn: **document.getElementsByTagName("tag")**

1. **getElementsByClassName()**

this methodreturns array of elements/tags which have same class name.

Syn: **document.getElementsByClassName("class")**

1. **querySelectorAll()**

this methodreturns array of elements/tags which are matching with specified selector.

we can use any CSS selectors:

> tag selector : tag

> ID selector : #id

> class selector : .classname

> grouping selector : tag1, tag2, ...

> child selector : parent-tag child-tag

Syn: **document.querySelectorAll("selector type")**

1. **querySelector()**

this methodreturns the first element/tag which are matching with specified selector.

Syn: **document.querySelector("selector type")**

1. **forms**

**element object**

**> this object represents single tag (it is not implicit object).**

**> element object used for interacting with tags/element for manipulations.**

**> means changing content of tag, for changing look & feel, we can add new tags/elements, we can delete existing element/tags etc...**

**> this "element" has some properties and methods.**

**tagName**

this property returns name of the tag/element.

Syn: **element-obj.tagName**

**id**

this property returns id of the tag.

Syn: **element-obj.id**

**innerHTML**

this property returns text/value/content of the tag and we can change text/value/content of tag.

Syn: **element-obj.innerHTML**

**element-obj.innerHTML="text"**

**paired & non-input🡺 innerHTML & innerText**

**for un paired tags & input 🡺 value**

**innerText**

this property returns text/value/content of the tag and we can change text/value/content of tag.

Syn: **element-obj.innerText**

**Value**

**style**

this property represents css style of the tag.

style is used to change css attributes and retrieve css attributes.

Syn: **element-obj.style.css-property**

**element-obj.style.css-property=newvalue;**

**<input type=”text” value=”10” maxlength=”3”>**

**setAttribute()**

this method is used to set/to add an attribute(html) to the existing tag.

Syn: **element-obj.setAttribute("att-name", "value")**

**Ele-obj.attribute=value;**

**getAttribute()**

this method returns the value of specified attribute(html) of the tag.

Syn: **element-obj.getAttribute("att-name")**

**Ele-obj.attribute**

**removeAttribute()**

this method is used to remove/delete the specified attribute(html) of the tag.

Syn: **element-obj.removeAttribute("att-name")**

**attributes**

this property returns all attributes(html) of a specified tag, along with values. it returns Collection object.

Syn: **element-obj.attributes**

**focus()**

this method places the cursor in requested element/control.

Syn: **element-obj.focus()**

**remove()**

this method used to remove the specified element/control/tag from current working webpage.

Syn: **element-obj.remove()**

**createElement()**

this method used to create a new element/control/tag (it manufactures element).

Syn: **let new-ele=document.createElement("tag-name");**

**appendChild()**

this method used to add new element/control as a child.

Syn: **element-obj.appendChild(new-ele)**

**window**

* **“window” object represents the current working browser window.**
* **browser creates only one window object.**
* **It has the following methods & properties**

**href**

this property represents url of the current working/running/viewing web page in browser window.

**window.href**

**window.href=newurl**

* + - **screenX**

this property represents X-cord of current browser position on the screen

Syn: **window.screenX**

**window.screenX=Npx**

* + - **screenY**

this property represents Y-cord of current browser position on the screen

**Syn: window.screenY**

**alert()**

**confirm()**

**prompt()**

**parseInt()**

**parseFloat()**

**setTimeout()**

call back, timer, it exe task, only once after n sec

var=setTimeout(task, “interval”)

**setInterval()**

call back, timer, it exe task, repeatedly for every n sec

var=setInterval(task, “interval”)

**clearTimeout() 🡺** it stops/clears the timeout function

clearTimeout(t-index)

**clearInterval() 🡺** it stops/clears the interval

clearInterval(t-index)

**print()**

this method displays print dialog box, which is used to print the current webpage/document through selected printer.

most of browser before printing page, they shows "Print Preview" also. **Syn: window.print()**