**DOM (Document Object Model)**

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

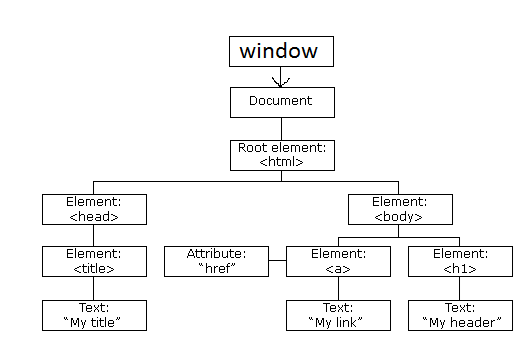
> When the web page has opened/loaded in browser, DOM will be automatically created by the browser.

>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".

Dom has the following detailed structure



The above structure represents as follows:

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

**Objects in DOM:**

* **window (this object providing interaction with browser window)**
* **document (this object providing interaction with webpage/webdoc)**
* **element (this object providing interaction with tags/elements)**

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

**object.propperty**

**object.method()**

**window**

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

1. **href**

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

Syn: **window**.**location.href**

1. **userAgent**

this property represents the name of current working browser.

Syn: **window.navigator.userAgent**

1. **Screen**
   * + **screenX**

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

Syn: **window.screenX**

* + - **screenY**

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

**Syn: window.screenY**

1. **alert()**
2. **confirm()**
3. **prompt()**
4. **setTimeout()**
5. **setInterval()**
6. **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()**

1. **open()**

this method opens a browser child window/pop window.

it is mainly useful for opening ads

**Syn: window.open("url", "title", "attribute=value, attribute=value, ...")**

1. **close()**

this method used for closing either child window/main window.

Syn: **window.close();**

**ref.close();**

1. **scrollTo()**

this method scrolls the webpage horizontally / vertically to the specified X and Y Co-ordinates.

the X and Y co-ordinates are calculated in the form of pixel.

Syn: **window.scrollTo(x, y)**

**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**

1. **head**

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

Syn: **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 representsall "images" of current web page, as an array format.

Syn: **document.images**

1. **link**

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

Syn: **document.links**

1. **url**

this property representsurl of current web page.

Syn: **document.URL**

1. **write()**

thismethod displays message in the web page.

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

1. **getElementById()**

thismethodreturns 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()**

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

thisspe used in checkbox and radio button case.

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

1. **getElementsByTagName()**

thismethodreturns array of elements/tags which have same tag name.

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

1. **getElementsByClassName()**

thismethodreturns array of elements/tags which have same class name.

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

1. **querySelectorAll()**

thismethodreturns 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()**

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

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

**element object**

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

1. **tagName**

thisproperty returns name of the tag/element.

Syn: **element-obj.tagName**

1. **id**

thisproperty returns id of the tag.

Syn: **element-obj.id**

1. **innerHTML**

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

Syn: **element-obj.innerHTML**

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

1. **innerText**

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

Syn: **element-obj.innerText**

1. **style**

thisproperty represents css style of the tag.

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

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

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

1. **parentElement**

thisproperty returnsparent element(tag) of current tag.

Syn: **element-obj.parentElement**

1. **children**

thisproperty returnschild element(tag) of current tag.

Syn: **element-obj.children**

1. **scrollTop**

thisproperty moves vertical scrollbar(adjusting data), based on the specified pixels.

Syn: **element-obj.scrollTop=no.ofpixels**

1. **setAttribute()**

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

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

1. **getAttribute()**

thismethod returns the value of specified attribute(html) of the tag.

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

1. **removeAttribute()**

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

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

1. **attributes**

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

Syn: **element-obj.attributes**

1. **hasAttribute()**

this method checks whether the element/tag has specified attribute or not, returns true if it contains, returns false if it doesn't contains.

Syn: **element-obj.hasAttribute("attribute-name")**

1. **focus()**

thismethod places the cursor in requested element/control.

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

1. **click()**

thismethod clicks the specified element/control automatically (means it's same as to manual mouse click).

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

1. **remove()**

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

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

1. **createElement()**

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

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

1. **appendChild()**

thismethod used to add new element/control as a child.

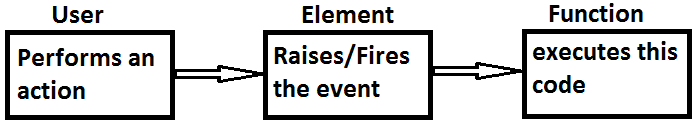
Syn: **element-obj.appendChild(new-ele)**

**Working with Events**

>event is a keyboard/mouse action, that is performed by user, at runtime.

> "event handling" is a concept of attaching the event with a function.

>whenever the user performs an action, automatically the element raises/fires the event, then we can call function.



**Syn: element-obj.addEventListener("event-name", fun-name/function-pointer)**

**List of Events**

javascript supports lot of events, those are:

1. click (when we click mouse left or right button one time)
2. dblclick (when we double click mouse left)
3. mouseover (whenever mouse comes top on element/control)
4. mouseout (whenever mouse comes out of element/control)
5. mousemove (if mouse moving on element/web page)
6. keyup (if any key releases in keyboard)
7. keypress (if any key pressing in keyboard)
8. focus (cursor entered)
9. blur (cursor leave)
10. change (if data changed in control)
11. contextmenu (if click on menu)
12. cut
13. copy
14. paste

etc...

<body>

<p id="demo">welocme to HTML Dom</p>

<script type='text/javascript'>

document.getElementById("demo").innerHTML="Hello world";

</script>

</body>

***Event.button***:It indicates which mouse button cause the event

syntax: event.button

w3c it's values should be:

* left --> button-0
* middle --> button-1
* right --> button -2

According to microsoft values should be:

* left --> button-1
* middle --> button-4
* Right --> button-2

ex:

<head>

<script type="text/javascript">

functionwhichButton(event) {

if(event.button==0) {

alert("you clicked the left mouse button!");

}

elseif(event.button==1) {

alert("you clicked the middle mouse button!");

}

elseif(event.button==2) {

alert("you clicked the right mouse button!");

}

else{

alert("event not identified");

}

}

</script>

</head>

<body onmousedown="whichButton(event)">

<p> Click Here observe...!</p>

</body>

***InnerHTMLproperty***:The inner HTML property is used along with getElementById within your javascript code to refer to an HTML element and change it's contents.

syntax: document.getElementById('{ID ofelement}').innerHTML='{context}';

**containers**: Element can hold other html elements/controls

ex: Div,p,Table,span...!

**Not containers**: Elements can hold only text can not hold htmlcontrols/Elements

ex: Textbox,Button,Radio,Textarea....!

**Note**: All containers are paired tag,But all paired tags are not containers (Containers having inner html property,non containers having valueproperty)

ex:

<head>

<script type='text/javascript'>

functionmytext(){

var value=document.getElementById('txt1').value;

alert("The value is:"+value);

}

</script>

</head>

<body>

<p>Click the button to dispaly the text from a text box based onvalue property...</p>

<input type='text' value="javascript" id="txt1"><br/>

<button onclick="mytext()">Clickme</button>

</body>

ex:

<head>

<script type='text/javascript'>

functionmytext(){

var value=document.getElementById('p1').innerHTML;

alert("The value is:"+value);

}

</script>

</head>

<body>

<p>Click the button to display the text from a text box based onvalue property...</p>

<p id="p1"><imgsrc="html5.png" width=100px height=100px></p>

<button onclick="mytext()">Clickme</button>

</body>

**Working with javascript validation**:

Javascript can be used to validate the data in html forms before sending of the containts to a server.Javascript form validation is provide a method to check a user entered information before click on submit. Form validation generaly perform in the following two ways

* Basic validation
* Data formate validation

**Basic validation**:The form must be checked to make sure data was enteredinto each form field that required it.This would need just loop througheach field in the form and check for data.

**Data formatevalidation**:The data that is entered must be checked forcorrect form and value.This would need to put more logic to testcorrectness of data.

**validatingTextBox**:

<head>

<script type='text/javascript'>

functionnotEmpty(){

varmyTextField=document.getElementById('myText');

if(myTextField.value!="") {

alert("you entered:"+myTextField.value);

}

else{

alert("would you please enter some text")

}

}

</script>

</head>

<body>

<input type='text' id='myText'/><br/>

<input type='button' onclick='notEmpty()' value='formValidation'/>

</body>

**Validation TextBox with border color**:

<head>

<script type='text/javascript'>

functionfunchklen(len,cid){

if(len<6) {

document.getElementById(cid).style.borderColor="red"

}

else{

document.getElementById(cid).style.borderColor="silver"

}

}

</script>

</head>

<body>

UserName:<input type="text" id="txt1" onblur="funchklen (this.value.length,'txt1')">

<br/>

Password:<input type="password" id="txt2" onblur="funchklen (this.value.length,'txt2')">

</body>

**Validating Radio Buttons**:

<head>

<script type="text/javascript">

function validate(){

var r1 =document.getElementById('male').checked;

var r2=document.getElementById(female).checked;

if((r1=="")&&(r2=="")){

alert("select either male or female");

return false;

}

return true;

}

</script>

</head>

<body>

<input type=radio id='male'/> male

<input type=radio id='female'/> female

<button onclick='validate()'>validate</button>

</body>

<head>

<script type='text/javascript'>

function fun1(){

if(document.getElementById('un').value.length>=6&&

document.getElementById('pw').value.length>=6){

document.getElementById('but1').disabled=false;

}

else{

document.getElementById('but1').disabled=true;

}

}

</script>

</head>

<body>

<input type='text' id='un' onblur='fun1()'>

<br/>

<input type='password' id='pw' onblur='fun1()'>

<br/>

<input type="button" value="click" id="but1" disabled>

</body>