DOM + Modern JS - 1

Window

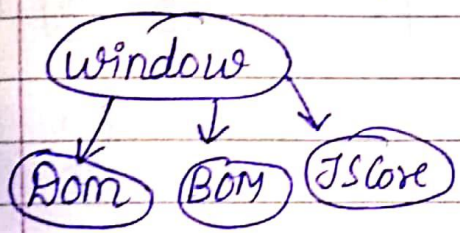
* Global object.
* Created by browser.
* Represents a browser window.
* Top Level Entity.

DOM (we'll study)

BOM

* Browser Object Model
* It allows JS to talk to Browser about matters other than content of Page.

ex:- location, History, Alert etc



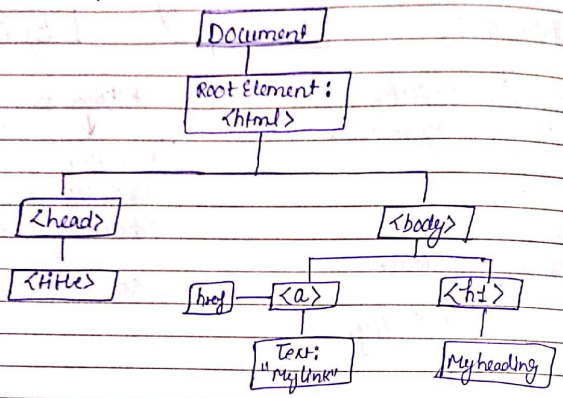
Ex:- Window.console.log(—)

DOM (Document Object Model)

It is tree like Structure.

character → tag → token → nodes → DOM

example of Tree like structure:-



With Object Model, :-

→ JS can change all the HTML elements, attributes, CSS styles in the page.

→ JS can add or remove HTML elements, attributes.

DOM Methods :- Accessing Elements

① getElementById :-

→ To access HTML Element by "id" of element.
→ It is called on document object.
→ It returns a single object because 'id' is always unique.

example :-
* document.getElementById('container')
* document.getElementById('Specifications')

② getElementsByClassName()

for multiple objects, The getElementByClassName method of Document interface returns an array-like object of all child elements which have all of the given classname(s).

document.getElementsByClassName('class-name')

example :-
document.getElementsByClassName('hidden').

③ getElementsByTagName() ← tagname

It returns an HTML collection of elements with the given tag name.

example :- * document.getElementsByTagName('p')
* document.getElementsByTagName('h1')

CLASSTEAM Page No. Date

* Keep in Mind :-

→ getElementsByClassName() & getElementsByTagName() both method use document object.

→ both return multiple items.
→ the list returned is not an array.

→ $0 returns the most recently selected element element or javascript object.

If we hover on anything or clicked, then go to Console type '$0' (dollar zero), we get Element of selected particular thing.

④ querySelector() :-

It returns the first element within the document that matches the specified selector, or group of selectors.
If no matches are found, null is returned.

examples :-

ⓐ ID Selector :- document.querySelector('#unique')

ⓑ Class Selector :- document.querySelector('.class')
↳ It returns only first class Element/tag

ⓒ Element Selector :- document.querySelector('H1'/'P'/HEADER')

Note :- If we want all class selector / multiple class selector
So, we use —

⑤ queryselectorAll() :-

ex :- document.queryselectorAll('.class')

It returns all elements with class name "class" multiple times objects.

Update Existing Content of Webpage.

→ .innerHTML (get/set)
→ .outerHTML
→ .textContent
→ .innerText

① .innerHTML :- * get an element or get all of its dependent elements.

* Set an element's HTML Content

example :-
```
let content = document.queryselector('.code example')
content.innerHTML
```
↓
Get all dependent tags with informations.

* `content.innerHTML = '';`

Here, Set the content in place of all descendents.

(2) .outerHTML :-

* Sets or returns the HTML element, including attributes, start tag and end tag.

Syntax :-

Return the outerHTML property —
`element.outerHTML`

Set the outerHTML property —
`element.outerHTML = text`

(3) .innerText :-

* This property Sets or returns the text content of an element.

Syntax :-

* `element.innerText` (Return)

or

* `element.innerText = text` (Set)

~~innerHTML~~

(4) .textContent :-

* This property sets or returns the text content of the specified node & all its descendants.

Syntax :-

* `element.textContent` (Return)

or

* `element.textContent = text` (Set)

Difference :-

innerHTML

  Bold

Rendering ✓

textContent

text

  text

Same as Text ✓

No Rendering ✗

Difference :-

textContent	innerText
<> — </>	<> — </>
<display : hidden> </>	
	* No Hidden text will come at starts.
* Hidden text will come .textContent at Starts	

Adding New Element / Content :-

(i) createElement() :- creates an element node.

Syntax :-
document.createElement(type)

example :-
~~let content = document.createElement~~

let content = document.querySelector('.paraclass');
content;

Output :- Data Structure

let newPara = document.createElement('H1');
newPara;

Output :- <H1> </H1>

(ii) appendChild() :- To append element created, at the end.

Ex :- content.appendChild(newPara);
content;

Output :- Data Structure <H1> </H1> //added New element.

Now, Empty

(iii) createTextNode() :- To add the textual content inside newly created child Element.

example :-
let newPara = document.createElement('p');
let textPara = document.createTextNode('I am NewPara');
newPara.appendChild(textPara);

Output :- I am New Para

example(2)

```
let textHead = document.createTextNode("I am Heading");
newPara.appendChild(textHead);
content.appendChild(newPara);
Content;
```

Output:

```
<p class="para class"> Data Structure
    <H1>" I am Heading" </H1>
</p>
```

* Alternate Method to add Content Inside Element.

example:-

```
let myPara = document.createElement('p');
myPara.textContent = 'I am my Para';
content.appendChild(myPara);
```

Output:

 I am my para .

(iv) insertAdjacentElement() :-

This method inserts Element into a specified position.

Syntax:-

element.insertAdjacentElement(position, element)

Positions are:-

afterbegin → After begin of the element (first child)
after end → After the element.
before begin → Before the element.
before end → Before the end of the element (last child)

---- before begin -----

---- After begin ------
 —
--- before End -----

---- After End -----

example:-

```
let content = $0;        → (current Element)
content;
```

 —
.

```
let newH = document.createElement('h3');
newH.textContent = 'Heading 3';

content.insertAdjacentElement('beforeend', newH);
```

Output:-

```
<p>
    <div> —— </div>

    <h3> Heading 3 </h3>
</p>
```

* → `content.insertAdjacentElement('Afterbegin', newH);`

Output:-

```
<p>
    <h3> Heading 3 </h3>
    <div> —— </div>
</p>
```

Removing Element :-

(i) • removeChild() :-

This Is the opposite of appendChild().

Syntax:- `parentElement.removeChild(childElement);`

Note:- We must have Parent Element & Child Element.

example:-

```
<p class = "parent">
    <h3 class = "child"> Heading 3 </h3>
    <div> </div>
</p>
```

```
let parent = document.querySelector('.parent');

let child = document.querySelector('.child');

parent.removeChild(child);
```

Output:- (after deleting <h3>)

```
<p>
    <div> —— </div>
</p>
```

(ii) Alternate Method :- `remove()`

```
let child = document.querySelector('.child');
child.parentElement.remove(child);
```

CSS Styling using JS DOM

- .style
- .cssText
- .setAttribute
- .className
- .classList

(i) **.style** :- style only 1 property at a time.

Syntax:- `element.style.property = value ;`

example:-

```
let content = $0 ;
content.style.color = 'red' ;
content.style.backgroundColor = 'white' ;
```

(ii) **.cssText** :- for multiple styling changes.

example :-

```
content.style.cssText = 'color: green ;
                         background-color : yellow ;
                         font-size : 4em ; ' ;
```

(iii) **.setAttribute** :- for styling change, can add Id & classes.

example :-

```
* content.setAttribute("style", "background-color : red ;");
* content.setAttribute("style", "color : orange ;
                                 background-color : green ;");
```

for adding Id / classes :-

```
* content.setAttribute("id", "Heading ID");
* content.setAttribute("class", "classname");
```

Drawback :- Separation of concern like for styling "style".
for Id → "id".
for class → "class".

(iv) **.className** :- This property sets or returns an element's classes. in string

examples :-

```
let content = $0 ;   // currently select element
content;
```

```
<div class="first second third">
</div>
```

`content.className ;`

Output:- 'first second third' → String

* for making array,

let classNames = content.className.split(' ');

output:-

['first', 'second', 'third'] → Array

(v.) .classList :-

* This property returns the CSS classnames of an element in the form of LIST (~~Array~~ Object)
* Return Array of classes → called DOMTokenLIST

example :-

content.classList ;

Ouput:-

DOMTokenLIST ['first', 'second', 'third']

methods available for classlist :-

- → add();
- → remove()
- → toggle() → toggle b/w tokens in list.
- → contains()
- → ~~remove()~~
- → length.

replace()
item().
etc...

(a) add() :- ~~content.add()~~

↳ content.classList.add('third');

['first', 'second', 'third']

(b) remove() :-

content.classList.remove('second');
content;

['first', 'third'].

(c) toggle() :-

content.classList.~~remove~~ toggle('fifth');
content;

['first', 'second', 'fifth'];

If classname absent, then Add

If classname Present, then Remove.

(d) contains() True/false of classnames.

content.classList.contains('first');

TRUE

(e) length → total no. of classes.

content.classList.length;

3