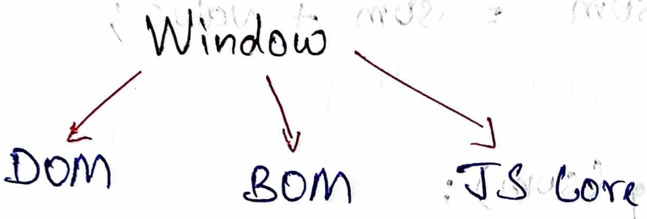
DOM + MODERN JS

Window

A window represents an open window in a browser.

It is a global object.

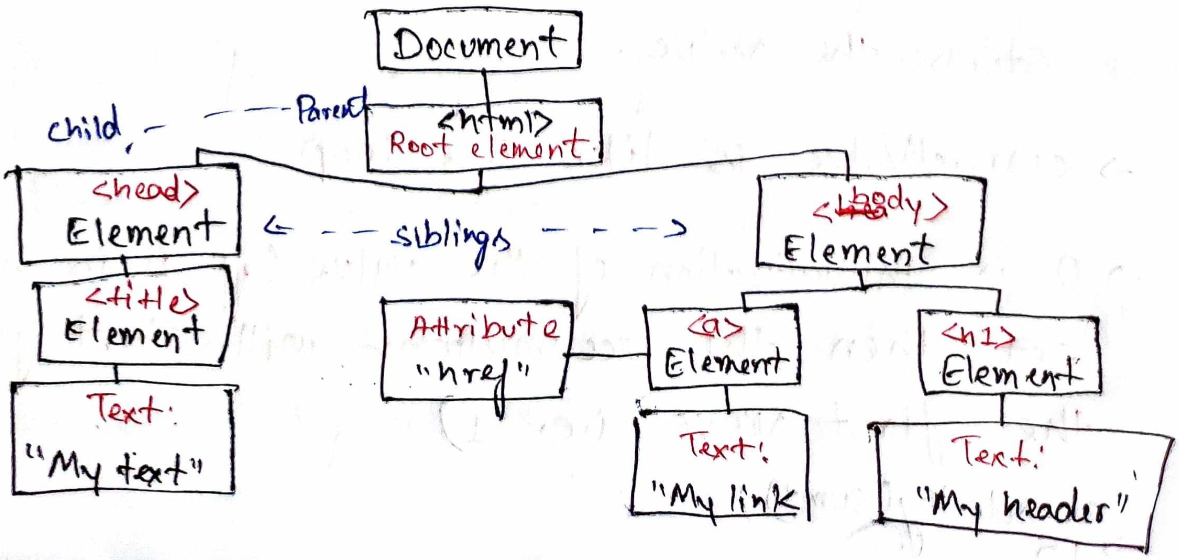
It is created by the browser



DOM (Document Object Model)

When a webpage is loaded, the browser creates a DOM of the page.

HTML DOM is a tree like structure



The HTML DOM Documents.

* The Document object

→ The doument object is the root node of the HTML document

→ The document object is a property of the window object

→ The document object is accessed with:

window.document or just document

Converting the whole HTML code into the JS object is called DOCUMENT and this Model is known as DOM.

Rendering of DOM

$$\begin{tikzcd}
\text{characters} \arrow[r] & \text{tags} \arrow[r, "\text{tokenized}"'] & \text{tokens} \arrow[r] & \text{nodes} \arrow[d] \\
& & & \text{DOM}
\end{tikzcd}$$

* DOM Methods (to access elements)

(Making changes in HTML using JS)

HTML DOM methods are actions you can perform (on HTML elements)

* The getElementById Method

The most common way to access an HTML element is to use the id of the element

Ex - getElementById('heading')

document.getElementById('content');

→ It is called on document object.

→ It returns a single object because it works on ID and its unique.

* The getElementsByClassName Method

This method of document interface returns an array-like object of all elements which have all the given class name(s).

NOTE - Array-like object is not an array.

Ex - document.getElementsByClassName("test");

* The getElementByTagName Method

→ It is similar to the previous method.

→ It also uses document object

→ It also returns multiple lines.

→ The list returned is not an array.

~~→~~ we can iterate on that list by using for loop.

Trick

$0 returns the most recently selected element or JavaScript object, $1 returns the second most recently selected one and so on.

* The querySelector Method

NOTE: Use querySelectorAll to get the multiple objects

This method returns the first element within the document that matches the specified selector, or group of selector.

If no match is ~~fun~~ found then null is returned.

Ex – querySelector('#header') → It is a ID
querySelector('.header') → It is a class
querySelector('header') → It is a tag

* Update Existing Content

* .innerHTML
* .outerHTML
* .textContent
* .innerText

* .innerHTML

→ It can get or set the ~~HM~~ HTML content within the element

→ It can get element or all of its descendents

→ It can set an element's HTML content

* .outerHTML

The outerHTML attribute of the element DOM interface gets the serialized HTML fragment describing the element including its descendants.

* .textContent

The textContent property sets or returns the text content of the specified node, and all its descendents.

Inner~~text~~ HTML vs Textcontent

In inner~~text~~ HTML if there is a tag inside a tag then it will be rendered but in case of textContent it will be treated as the normal text.

Ex.
```
<p>
    <b> SWE </b>
</p>
```

⇒ In case of inner~~Text~~ HTML the 'SWE' will be bold. **SWE**

⇒ In case of textContent `<b>` will be printed as it is without bolding the SWE.
`<b> SWE </b>`

* .innerText

It is similar to .textContent with some difference.

Ex - <p>

```
<> —— <>
<display:hidden> —— <>
```

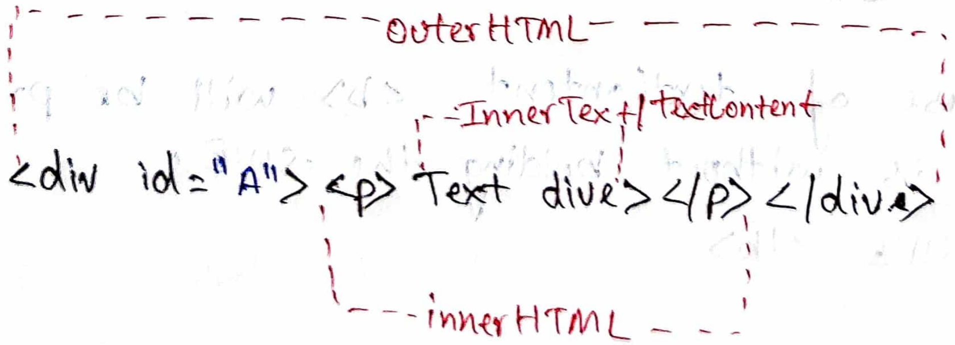
</p>

⇒ In case of text content the selected part will be printed.

⇒ But if we use innerText the selected part will be printed except the display: hidden line.

⇒ If the display: hidden property is used then it will not appear when we use innerText.

Visual Representation



```
<div id="A"><p> Text div's</p></div>
```

* Adding new content / element

To add a content we need to first create the content.

* .createElement()

This method creates the HTML element specified by tagName.

Syntax.

createElement(tagName)

⇒ .appendChild()

→ To add the content we use .appendChild().

→ It puts the content at the last position in the element.

* Creating a Text Node

It's a long way

```
let newPara = document.createElement('p');
let textPara = document.createTextNode('Hello jee');
newPara.appendChild(textPara);
content.appendChild(newPara);
```

<p> </p> ⟶ <p> Hello jee </p>

Short way

```
let myPara = document.createElement('p');
myPara.textcontent = 'Hello jee';
content.appendChild(myPara);
```

From the above tag we can ~~of~~ only add siblings at the last position only.

To add the sibling at the specified position we use below method.

* .insertAdjacentHTML()

It has to be called with 2 arguements

i) location / position → where

ii) HTML text / content → what

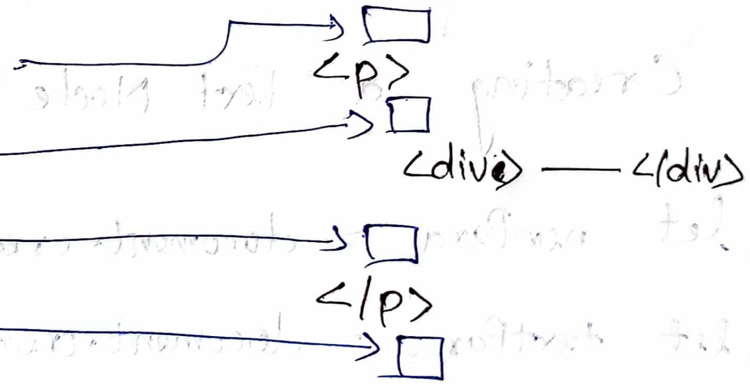
There are 4 positions in it:

* beforebegin

* afterbegin

* beforeend

* afterent



* Deleting an Element / Content

⇒ .removeChild()

→ It is opposite of .appendChild()

→ We must know its parent element.

→ We must know the child element which we are removing.

Syntax

```
parent.removeChild(cElement);
```

Deleting a child element without knowing its parent element.

Here, Parent = childElement.parent

```
child.parent.remove(child);
```

* Making changes in CSS using JS

* .style → you can make single change only
* .CSSText
* .setAttribute
* .className
* .classList

(.CSSText, .setAttribute, .className, .classList) you can make multiple changes at a time.

Ex let content = $0;

```
→ content.style.color = 'red';
→ content.cssText = 'color:red, font-size:4em;';
→ content.setAttribute("style","color:red; background-
                       color: white;");
```

* .className

→ It sets or returns an element's class attribute.

→ It returns all the class names as a string and to make change you'll have to convert it into an array.

* .classList

→ It returns the CSS classnames of an element.

→ It returns in the array format of 'object' type.

Features

add()	– to add ~~an~~ class
remove()	– to remove ~~as~~ class
toggle()	– if class is present then it will delete it if not present then it will add it.
contains()	– to check whether a class class is present or not. It returns True or false.