The document model

The document object model is data representation if the objects that comprise the structure and content of a document on the web

The document object model is the JavaScript model of a webpage.it contains representation of all the content on a page.

Selecting the elements

* getElementById
* getElementsByTagName
* getElementsByClassName

getElementById:

he [Document](https://developer.mozilla.org/en-US/docs/Web/API/Document) method **getElementById()** returns an [Element](https://developer.mozilla.org/en-US/docs/Web/API/Element) object representing the element whose [id](https://developer.mozilla.org/en-US/docs/Web/API/Element/id) property matches the specified string.

Syntax:

Var element = document.getElementById(id);

getElementByTagName:

The **getElementsByTagName** method of [Document](https://developer.mozilla.org/en-US/docs/Web/API/Document) interface returns an [HTMLCollection](https://developer.mozilla.org/en-US/docs/Web/API/HTMLCollection) of elements with the given tag name.

Syntax:

Var element = document.getElementByTagName(name);

getElementByClassName:

The **getElementsByClassName** method of [Document](https://developer.mozilla.org/en-US/docs/Web/API/Document) interface returns an array-like object of all child elements which have all of the given class name(s).

Syntax:

Var elements = document.getElementByClassName(name);

Query selectors:

The [Document](https://developer.mozilla.org/en-US/docs/Web/API/Document) method **querySelector()** returns the first [Element](https://developer.mozilla.org/en-US/docs/Web/API/Element) within the document that matches the specified selector, or group of selectors. If no matches are found, null is returned.

Syntax:

Element = document.querySelector(selectors);

Element.querySelector():

The **querySelector()** method of the [Element](https://developer.mozilla.org/en-US/docs/Web/API/Element) interface returns the first element that is a descendant of the element on which it is invoked that matches the specified group of selectors.

Syntax:

Element = baseElement.querySelector(selectors);

Document.querySelectorAll(),Element.querySelectorAll():

The [Document](https://developer.mozilla.org/en-US/docs/Web/API/Document) method **querySelectorAll()** returns a static (not live) [NodeList](https://developer.mozilla.org/en-US/docs/Web/API/NodeList) representing a list of the document's elements that match the specified group of selectors.

Syntax:

elementList =parentNode.querySelectorAll(Selectors);

List of important properties and methods:

* classList

The **Element.classList** is a read-only property that returns a live [DOMTokenList](https://developer.mozilla.org/en-US/docs/Web/API/DOMTokenList) collection of the class attributes of the element. This can then be used to manipulate the class list.

Syntax:

Const elementClasses = elementNodeReference.classList;

* getAttribute():

The **getAttribute()** method of the [Element](https://developer.mozilla.org/en-US/docs/Web/API/Element) interface returns the value of a specified attribute on the element.

Syntax:

Let attribute = element,getAttributes(AttrivuteName);

* setAttribute()

Sets the value of an attribute on the specified element. If the attribute already exists, the value is updated; otherwise a new attribute is added with the specified name and value.

Syntax:

Element.setAttribute(name, value);

* appendChild()

The **Node.appendChild()** method adds a node to the end of the list of children of a specified parent node. If the given child is a reference to an existing node in the document, appendChild() moves it from its current position to the new position (there is no requirement to remove the node from its parent node before appending it to some other node).

Syntax:

Element.appendchild(aChild)

* append()

The **Element.append()** method inserts a set of [Node](https://developer.mozilla.org/en-US/docs/Web/API/Node) objects or [DOMString](https://developer.mozilla.org/en-US/docs/Web/API/DOMString) objects after the last child of the Element. [DOMString](https://developer.mozilla.org/en-US/docs/Web/API/DOMString) objects are inserted as equivalent [Text](https://developer.mozilla.org/en-US/docs/Web/API/Text) nodes.

Syntax:

Append(…nodes)

* prepend()

The **Element.prepend()** method inserts a set of [Node](https://developer.mozilla.org/en-US/docs/Web/API/Node) objects or [DOMString](https://developer.mozilla.org/en-US/docs/Web/API/DOMString) objects before the first child of the [Element](https://developer.mozilla.org/en-US/docs/Web/API/Element)

Syntax:

Prepend(…stings)

* removeChild()

The **Element.prepend()** method inserts a set of [Node](https://developer.mozilla.org/en-US/docs/Web/API/Node) objects or [DOMString](https://developer.mozilla.org/en-US/docs/Web/API/DOMString) objects before the first child of the [Element](https://developer.mozilla.org/en-US/docs/Web/API/Element)

Syntax:

Var oldChild = node.removeChild(child);

* remove()

The **Element.remove()** method removes the element from the tree it belongs to.

Syntax:

Remove()

* createElement

In an [HTML](https://developer.mozilla.org/en-US/docs/Web/HTML) document, the **document.createElement()** method creates the HTML element specified by tagName, or an [HTMLUnknownElement](https://developer.mozilla.org/en-US/docs/Web/API/HTMLUnknownElement) if tagName isn't recognized.

Syntax:

Let element = document.createElement(tagName[, option]);

* innerText

The **innerText** property of the [HTMLElement](https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement) interface represents the "rendered" text content of a node and its descendants.

Syntax:

const renderedText = htmlElement.innerText

htmlElement.innerText = string

* textContent
* innerHTML
* value
* parentElement
* children
* nextSibling
* previousSibling
* style

Events:

Events are actions or occurrences that happen in the system you are programming, which the system tells you about so you can respond to them in some way if desired.

Events:

* clicks

An element receives a **click** event when a pointing device button (such as a mouse's primary mouse button) is both pressed and released while the pointer is located inside the element.

* Drags

The drag event is fired every few hundred milliseconds as an element or text selection is being dragged by the user.

* Drops

The **drop** event is fired when an element or text selection is dropped on a valid drop target.

* Hovers

The **mouseover** event is fired at an [Element](https://developer.mozilla.org/en-US/docs/Web/API/Element) when a pointing device (such as a mouse or trackpad) is used to move the cursor onto the element or one of its child elements.

* Scrolls

The **scroll()** method of the [Element](https://developer.mozilla.org/en-US/docs/Web/API/Element) interface scrolls the element to a particular set of coordinates inside a given element.

Syntax:

Element.scroll(options)

* Form
* HTML form validation can be done by JavaScript.
* If a form field (fname) is empty, this function alerts a message, and returns false, to prevent the form from being submitted:
* submission
* keypresses
* focus
* mouse wheel
* double click
* copying
* pasting
* audio start
* screen size
* printing

**EventListener**

The **EventListener** interface represents an object that can handle an event dispatched by an [EventTarget](https://developer.mozilla.org/en-US/docs/Web/API/EventTarget) object.