=====================DOM====================

Browser Environment:
=====================

*The JavaScript language was initially created for web browsers.
   Since then, it has evolved into a language with many uses and platforms.
*A platform may be a browser, or a web-server or another host.
*A host environment provides its own objects and functions in addition to the language core.
           Web browsers give a means to control web pages. Node.js provides server-side features

 ========>JavaScript runs in a web browser<===========

```
                 window
                                    |
                                        |
        DOM            BOM         JavaScript

         documnet
```

*There’s a “root” object called window

  ===>this window represent "Global Object".
  ====> represents the “browser window”

ex:

```
function sayHi() {
  alert("Hello");
}

// global functions are methods of the global object:
window.sayHi();


alert(window.innerHeight); // inner window height
```

==============================DOM(Document Object Model)=================================

*The Document Object Model, or DOM for short, represents all page content as objects that can be modified.
*The document object is the main “entry point” to the page.
*We can change or create anything on the page using dom.

ex:

```
// change the background color to red
document.body.style.background = "red";

// change it back after 1 second
setTimeout(() => document.body.style.background = "", 1000);
```

===BOM(Browser Object model)=========================

*The Browser Object Model (BOM) represents additional objects provided by the browse for working with everything except the document.

**the two most widely known are: navigator.userAgent
ex:
--

```
alert(location.href); // shows current URL
if (confirm("Go to Wikipedia?")) {
  location.href = "https://wikipedia.org"; // redirect the browser to another URL
}
```

=>The functions alert/confirm/prompt are also a part of the BOM: they are not directly related to the document,
          but represent pure browser methods for communicating with the user.

Dom Tree:
=========

=>The backbone of an HTML document is tags.
=>According to the Document Object Model (DOM), every HTML tag is an object.
=>Nested tags are “children” of the enclosing one. The text inside a tag is an object as well.
=>*All these objects are accessible using JavaScript, and we can use them to modify the page.

```
ex:  document.body  = <body> tag   // both are equal
```

**All operations on the DOM start with the document object.
        "That’s the main “entry point” to DOM"

The topmost tree nodes are available directly as document properties:

<html> = document.documentElement
The topmost document node is document.documentElement. That’s the DOM node of the <html> tag.

<body> = document.body
Another widely used DOM node is the <body> element – document.body.

<head> = document.head
The <head> tag is available as document.head.

Children: childNodes, firstChild, lastChild:
===

ex:
---->

```
<html>
<body>
  <div>Begin</div>

  <ul>
    <li>Information</li>
  </ul>

  <div>End</div>

  <script>
    for (let i = 0; i < document.body.childNodes.length; i++) {
      alert( document.body.childNodes[i] ); // Text, DIV, Text, UL, ..., SCRIPT
    }
  </script>
  ...more stuff...
</body>
</html>
```

For example:
============

```
// parent of <body> is <html>
alert( document.body.parentNode === document.documentElement ); // true
```

```
// after <head> goes <body>
alert( document.head.nextSibling ); // HTMLBodyElement

// before <body> goes <head>
alert( document.body.previousSibling ); // HTMLHeadElement
```

==
Searching: getElement*, querySelector*:
==

document.getElementById :
----

*If an element has the id attribute, we can get the element using the method " document.getElementById(id) "

ex:

```
<body>
<div id="elem">
  <div id="elem-content">Element</div>
</div>

<script>
  // get the element
  let elem = document.getElementById('elem');

  // make its background red
  elem.style.background = 'red';
</script>

</body>
```

The id must be unique:

=>The id must be unique. There can be only one element in the document with the given id.
=>If there are multiple elements with the same id, then the behavior of methods that use it is unpredictable.
       eg: document.getElementById may return any of such elements at random.

querySelectorAll:
=================

=>he most versatile method, elem.querySelectorAll(css) returns all elements inside elem matching the given CSS selector.

ex:
---->

```
<ul>
  <li>The</li>
  <li>test</li>
</ul>
<ul>
  <li>has</li>
  <li>passed</li>
</ul>
<script>
  let elements = document.querySelectorAll('ul > li:last-child');

  for (let elem of elements) {
    alert(elem.innerHTML); // "test", "passed"
  }
</script>
```

*This method is indeed powerful, because any CSS selector can be used.

querySelector:
===============

=>The call to elem.querySelector(css) returns the first element for the given CSS selector.

=>the result is the same as elem.querySelectorAll(css)[0], but the latter is looking for all elements and picking one,
                  while elem.querySelector just looks for one. So it’s faster and also shorter to write.

ex:
---->

```
<a href="http://example.com/file.zip">...</a>
<a href="http://ya.ru">...</a>

<script>
  // can be any collection instead of document.body.children
  for (let elem of document.body.children) {
    if (elem.matches('a[href$="zip"]')) {
      alert("The archive reference: " + elem.href );
    }
  }
</script>
```

closest:
========

=>Ancestors of an element are: parent, the parent of parent, its parent and so on.
            The ancestors together form the chain of parents from the element to the top.
=>The method elem.closest(css) looks for the nearest ancestor that matches the CSS-selector.
           The elem itself is also included in the search.
=>**the method closest goes up from the element and checks each of parents.
            If it matches the selector, then the search stops, and the ancestor is returned.

ex:
--->

```
<h1>Contents</h1>

<div class="contents">
  <ul class="book">
    <li class="chapter">Chapter 1</li>
    <li class="chapter">Chapter 2</li>
  </ul>
</div>

<script>
  let chapter = document.querySelector('.chapter'); // LI

  alert(chapter.closest('.book')); // UL
  alert(chapter.closest('.contents')); // DIV

  alert(chapter.closest('h1')); // null (because h1 is not an ancestor)
</script>
```

******getElementsBy*=======>
=============================

=>There are also other methods to look for nodes by a tag, class, etc.

=>They are mostly history, as querySelector is more powerful and shorter to write.

other methods of: getElementsBy*:

=> elem.getElementsByTagName(tag) looks for elements with the given tag and returns the collection of them.
       The tag parameter can also be a star "*" for “any tags”.
=>elem.getElementsByClassName(className) returns elements that have the given CSS class.

=>document.getElementsByName(name) returns elements with the given name attribute, document-wide. Very rarely used

****************************Don’t forget the "s" letter!

ex:

```
// get all divs in the document
let divs = document.getElementsByTagName('div');
```

ex:
----->

ind all input tags inside the table:

```
<table id="table">
  <tr>
    <td>Your age:</td>

    <td>
      <label>
        <input type="radio" name="age" value="young" checked> less than 18
      </label>
      <label>
        <input type="radio" name="age" value="mature"> from 18 to 50
      </label>
      <label>
        <input type="radio" name="age" value="senior"> more than 60
      </label>
    </td>
  </tr>
</table>

<script>
  let inputs = table.getElementsByTagName('input');

  for (let input of inputs) {
    alert( input.value + ': ' + input.checked );
  }
</script>
```

ex 2:
------>

```
<form name="my-form">
  <div class="article">Article</div>
  <div class="long article">Long article</div>
</form>

<script>
  // find by name attribute
  let form = document.getElementsByName('my-form')[0];

  // find by class inside the form
  let articles = form.getElementsByClassName('article');
  alert(articles.length); // 2, found two elements with class "article"
</script>
```

LIve collections:
----

*To overcome getementsbyclass names
   ** querySelectorAll returns a static collection. It’s like a fixed array of elements.

*****querySelector is better than other methods(getElementsBy*)

ex:

```
<div>First div</div>

<script>
  let divs = document.querySelectorAll('div');
  alert(divs.length); // 1
</script>

<div>Second div</div>

<script>
  alert(divs.length); // 1
</script>
```

***There are 6 main methods to search for nodes in DOM:

Method	Searches by...
querySelector	CSS-selector
querySelectorAll	cSS-selector
getElementById	id
getElementsByName	name
getElementsByTagName	tag or '*'
getElementsByClassName	class

================================
Attributes and properties:::::
================================

HTML attributes:

=>When the browser parses the HTML to create DOM objects for tags, it recognizes standard attributes and creates DOM properties from them.

ex:

```
<body id="test" something="non-standard">
  <script>
    alert(document.body.id); // test
    // non-standard attribute does not yield a property
    alert(document.body.something); // undefined
  </script>
</body>
```

 **********Please note that a standard attribute for one element can be unknown for another one
   ex:

```
   <body id="body" type="...">
  <input id="input" type="text">
  <script>
    alert(input.type); // text
    alert(body.type); // undefined: DOM property not created, because it's
non-standard
  </script>
</body>
```

*** if an attribute is non-standard, there won’t be a DOM-property for it

All attributes are accessible by using the following methods:
----

elem.hasAttribute(name) – checks for existence.
elem.getAttribute(name) – gets the value.
elem.setAttribute(name, value) – sets the value.
elem.removeAttribute(name) – removes the attribute.

HTML attributes have the following features:

Their name is case-insensitive (id is same as ID).
Their values are always strings.

ex:

```
<body>
  <div id="elem" about="Elephant"></div>

  <script>
    alert( elem.getAttribute('About') ); // (1) 'Elephant', reading

    elem.setAttribute('Test', 123); // (2), writing

    alert( elem.outerHTML ); // (3), see if the attribute is in HTML (yes)

    for (let attr of elem.attributes) { // (4) list all
      alert( `${attr.name} = ${attr.value}` );
    }
  </script>
</body>
```

DOM properties are typed:

DOM properties are not always strings.
  **For instance, the input.checked property (for checkboxes) is a boolean:

ex:
--

```
<input id="input" type="checkbox" checked> checkbox

<script>
  alert(input.getAttribute('checked')); // the attribute value is: empty string
  alert(input.checked); // the property value is: true
</script>


===========================
Modifying the document:
========================


=>DOM modification is the key to creating “live” pages.


Creating an element:
----------------------
      To create DOM nodes, there are two methods:
          -----------------------------------------
          1.document.createElement(tag)
               ex:
                    ---
                        Creates a new element node with the given tag:

                          let div = document.createElement('div');

     2.document.createTextNode(text)
          ex:
          ---
            Creates a new text node with the given text:

                          let textNode = document.createTextNode('Here I am');


Creating the message div takes 3 steps:
----------------------------------------

// 1. Create <div> element
let div = document.createElement('div');

// 2. Set its class to "alert"
div.className = "alert";

// 3. Fill it with the content
div.innerHTML = "<strong>Hi there!</strong> You've read an important message.";

  ***But as of now it’s only in a variable named div, not in the page yet ryt?

Insertion methods:
------------------
   *To make the div show up, we need to insert it somewhere into document
   *For instance, into <body> element, referenced by document.body.
   *****There’s a special method append for that:
```

```
document.body.append(div).


   ex:---
   ---------

   <style>
.alert {
  padding: 15px;
  border: 1px solid #d6e9c6;
  border-radius: 4px;
  color: #3c763d;
  background-color: #dff0d8;
}
</style>

<script>
  let div = document.createElement('div');
  div.className = "alert";
  div.innerHTML = "<strong>Hi there!</strong> You've read an important message.";

  document.body.append(div);
</script>




==>Here we called append on document.body,
     but we can call append method on any other element, to put another element into
it.


Here are more insertion methods, they specify different places where to insert:
------------------------------------------------------------------------------

==>node.append(...nodes or strings) – append nodes or strings at the end of node,
==>node.prepend(...nodes or strings) – insert nodes or strings at the beginning of
node,
==>node.before(...nodes or strings) -- insert nodes or strings before node,
==>node.after(...nodes or strings) –- insert nodes or strings after node,
==>node.replaceWith(...nodes or strings) -- replaces node with the given nodes or
strings.


example of using these methods to add items to a list and the text before/after it:
---------------------

ex:
---

<ol id="ol">
```

```
  <li>0</li>
  <li>1</li>
  <li>2</li>
</ol>

<script>
  ol.before('before'); // insert string "before" before <ol>
  ol.after('after'); // insert string "after" after <ol>

  let liFirst = document.createElement('li');
  liFirst.innerHTML = 'prepend';
  ol.prepend(liFirst); // insert liFirst at the beginning of <ol>

  let liLast = document.createElement('li');
  liLast.innerHTML = 'append';
  ol.append(liLast); // insert liLast at the end of <ol>
</script>
```

final list wil be:

```
before
<ol id="ol">
  <li>prepend</li>
  <li>0</li>
  <li>1</li>
  <li>2</li>
  <li>append</li>
</ol>
after
```

Node removal:

To remove a node, there’s a method node.remove().

Let’s make our message disappear after a second:

ex:

```
<style>
.alert {
  padding: 15px;
  border: 1px solid #d6e9c6;
  border-radius: 4px;
  color: #3c763d;
  background-color: #dff0d8;
}
</style>
```

```
<script>
  let div = document.createElement('div');
  div.className = "alert";
  div.innerHTML = "<strong>Hi there!</strong> You've read an important message.";

  document.body.append(div);
  setTimeout(() => div.remove(), 1000);
</script>
```

Please note: if we want to move an element to another place – there's no need to remove it from the old one.

Methods to create new nodes:
=============================

document.createElement(tag) – creates an element with the given tag,
document.createTextNode(value) – creates a text node (rarely used),
elem.cloneNode(deep) – clones the element, if deep==true then with all descendants.
Insertion and removal:

node.append(...nodes or strings) – insert into node, at the end,
node.prepend(...nodes or strings) – insert into node, at the beginning,
node.before(...nodes or strings) -- insert right before node,
node.after(...nodes or strings) -- insert right after node,
node.replaceWith(...nodes or strings) -- replace node.
node.remove() -- remove the node.
Text strings are inserted "as text".

There are also "old school" methods:

parent.appendChild(node)
parent.insertBefore(node, nextSibling)
parent.removeChild(node)
parent.replaceChild(newElem, node)
All these methods return node.

Given some HTML in html, elem.insertAdjacentHTML(where, html) inserts it depending on the value of where:

"beforebegin" – insert html right before elem,
"afterbegin" – insert html into elem, at the beginning,
"beforeend" – insert html into elem, at the end,
"afterend" – insert html right after elem.
Also there are similar methods, elem.insertAdjacentText and elem.insertAdjacentElement, that insert text strings and elements, but they are

rarely used.

To append HTML to the page before it has finished loading:

document.write(html)
After the page is loaded such a call erases the document. Mostly seen in old scripts.

=====================
Styles and classes:
====================

There are generally two ways to style an element:

1.Create a class in CSS and add it: <div class="...">
2.Write properties directly into style: <div style="...">.

 **JavaScript can modify both classes and style properties.

=>For example, style is acceptable if we calculate coordinates of an element dynamically and want to set them from JavaScript, like this:

ex:
--
```
let top = /* complex calculations */;
let left = /* complex calculations */;

elem.style.left = left; // e.g '123px', calculated at run-time
elem.style.top = top; // e.g '456px'
```

To manage classes, there are two DOM properties:
----
className – the string value, good to manage the whole set of classes.
classList – the object with methods add/remove/toggle/contains, good for individual classes.
To change the styles:

The style property is an object with camelCased styles. Reading and writing to it has the same meaning as modifying individual properties in the "style" attribute. To see how to apply important and other rare stuff – there’s a list of methods at MDN.

The style.cssText property corresponds to the whole "style" attribute, the full string of styles.

To read the resolved styles (with respect to all classes, after all CSS is applied and final values are calculated):

The getComputedStyle(elem, [pseudo]) returns the style-like object with them.
Read-only.

===========================
Element size and scrolling:
=============================

*There are many JavaScript properties that allow us to read information about element width, height and other geometry features.

Elements have the following geometry properties:
----
offsetParent – is the nearest positioned ancestor or td, th, table, body.
offsetLeft/offsetTop – coordinates relative to the upper-left edge of offsetParent.
offsetWidth/offsetHeight – “outer” width/height of an element including borders.
clientLeft/clientTop – the distances from the upper-left outer corner to the upper-left inner (content + padding) corner. For left-to-right OS they are always the widths of left/top borders. For right-to-left OS the vertical scrollbar is on the left so clientLeft includes its width too.
clientWidth/clientHeight – the width/height of the content including paddings, but without the scrollbar.
scrollWidth/scrollHeight – the width/height of the content, just like clientWidth/clientHeight, but also include scrolled-out, invisible part of the element.
scrollLeft/scrollTop – width/height of the scrolled out upper part of the element, starting from its upper-left corner.

=================================
Window sizes and scrolling::::
=================================

==>How do we find the width and height of the browser window? How do we get the full width and height of the document,
 including the scrolled out part? .
   ==>How do we scroll the page using JavaScript?

ex1:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JS Select Topmost Elements</title>
</head>
```

```
<body>
    <script>
    // Display lang attribute value of html element
    alert(document.documentElement.getAttribute("lang")); // Outputs: en

    // Set background color of body element
    document.body.style.background = "yellow";

    // Display tag name of the head element's first child
    alert(document.head.firstElementChild.nodeName); // Outputs: meta
    </script>
</body>
</html>


ex2:
-----

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JS Document.body Demo</title>
    <script>
    alert("From HEAD: " + document.body); // Outputs: null (since <body> is not
parsed yet)
    </script>
</head>
<body>
    <script>
    alert("From BODY: " + document.body); // Outputs: HTMLBodyElement
    </script>
</body>
</html>

ex3:
--

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JS Select Element by ID</title>
</head>
<body>
    <p id="mark">This is a paragraph of text.</p>
    <p>This is another paragraph of text.</p>

    <script>
    // Selecting element with id mark
    var match = document.getElementById("mark");

    // Highlighting element's background
    match.style.background = "yellow";
    </script>
</body>
</html>
```

ex4:
--

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JS Select Elements by Class Name</title>
</head>
<body>
    <p class="test">This is a paragraph of text.</p>
    <div class="block test">This is another paragraph of text.</div>
    <p>This is one more paragraph of text.</p>

    <script>
    // Selecting elements with class test
    var matches = document.getElementsByClassName("test");

    // Displaying the selected elements count
    document.write("Number of selected elements: " + matches.length);

    // Applying bold style to first element in selection
    matches[0].style.fontWeight = "bold";

    // Applying italic style to last element in selection
    matches[matches.length - 1].style.fontStyle = "italic";

    // Highlighting each element's background through loop
    for(var elem in matches) {
        matches[elem].style.background = "yellow";
    }
    </script>
</body>
</html>
```

ex6:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JS Select Elements by Tag Name</title>
</head>
```

```
<body>
    <p>This is a paragraph of text.</p>
    <div class="test">This is another paragraph of text.</div>
    <p>This is one more paragraph of text.</p>

    <script>
    // Selecting all paragraph elements
    var matches = document.getElementsByTagName("p");

    // Printing the number of selected paragraphs
    document.write("Number of selected elements: " + matches.length);

    // Highlighting each paragraph's background through loop
    for(var elem in matches) {
        matches[elem].style.background = "yellow";
    }
    </script>
</body>
</html>


ex:
---
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JS Select Elements with CSS Selectors</title>
</head>
<body>
    <ul>
        <li>Bread</li>
        <li class="tick">Coffee</li>
        <li>Pineapple Cake</li>
    </ul>

    <script>
    // Selecting all li elements
    var matches = document.querySelectorAll("ul li");

    // Printing the number of selected li elements
    document.write("Number of selected elements: " + matches.length + "<hr>")

    // Printing the content of selected li elements
    for(var elem of matches) {
        document.write(elem.innerHTML + "<br>");
    }

    // Applying line through style to first li element with class tick
    matches = document.querySelectorAll("ul li.tick");
    matches[0].style.textDecoration = "line-through";
    </script>
</body>
</html>


JavaScript DOM Styling:
----------------------

ex:
--<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JS Set Inline Styles Demo</title>
</head>
<body>
    <p id="intro">This is a paragraph.</p>
    <p>This is another paragraph.</p>

    <script>
    // Selecting element
    var elem = document.getElementById("intro");

    // Appling styles on element
    elem.style.color = "blue";
    elem.style.fontSize = "18px";
    elem.style.fontWeight = "bold";
    </script>
</body>
</html>

ex:
--

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JS Get Element's Style Demo</title>
</head>
<body>
    <p id="intro" style="color:red; font-size:20px;">This is a paragraph.</p>
    <p>This is another paragraph.</p>

    <script>
    // Selecting element
    var elem = document.getElementById("intro");

    // Getting style information from element
    alert(elem.style.color);  // Outputs: red
    alert(elem.style.fontSize);  // Outputs: 20px
    alert(elem.style.fontStyle);  // Outputs nothing
    </script>
</body>
</html>
```

ex:
--

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>JS Get Computed Style Demo</title>
<style type="text/css">
    #intro {
        font-weight: bold;
        font-style: italic;
    }
</style>
</head>
<body>
    <p id="intro" style="color:red; font-size:20px;">This is a paragraph.</p>
    <p>This is another paragraph.</p>

    <script>
    // Selecting element
    var elem = document.getElementById("intro");

    // Getting computed style information
    var styles = window.getComputedStyle(elem);

    alert(styles.getPropertyValue("color"));  // Outputs: rgb(255, 0, 0)
    alert(styles.getPropertyValue("font-size"));  // Outputs: 20px
    alert(styles.getPropertyValue("font-weight"));  // Outputs: 700
    alert(styles.getPropertyValue("font-style"));  // Outputs: italic
    </script>
</body>
</html>
```

ex:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>JS Add or Replace CSS Classes Demo</title>
<style>
    .highlight {
        background: yellow;
    }
```

```
</style>
</head>
<body>
    <div id="info" class="disabled">Something very important!</div>

    <script>
    // Selecting element
    var elem = document.getElementById("info");

    elem.className = "note";  // Add or replace all classes with note class
    elem.className += " highlight";  // Add a new class highlight
    </script>
</body>
</html>
```

ex:
--

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>JS classList Demo</title>
<style>
    .highlight {
        background: yellow;
    }
</style>
</head>
<body>
    <div id="info" class="disabled">Something very important!</div>

    <script>
    // Selecting element
    var elem = document.getElementById("info");

    elem.classList.add("hide");  // Add a new class
    elem.classList.add("note", "highlight");  // Add multiple classes
    elem.classList.remove("hide"); // Remove a class
    elem.classList.remove("disabled", "note"); // Remove multiple classes
    elem.classList.toggle("visible"); // If class exists remove it, if not add it

    // Determine if class exist
    if(elem.classList.contains("highlight")) {
        alert("The specified class exists on the element.");
    }
    </script>
</body>
</html>
```

JavaScript DOM Get Set Attributes:

```
----------------------------

ex:
---
<a href="https://www.google.com/" target="_blank" id="myLink">Google</a>

<script>
    // Selecting the element by ID attribute
    var link = document.getElementById("myLink");

    // Getting the attributes values
    var href = link.getAttribute("href");
    alert(href); // Outputs: https://www.google.com/

    var target = link.getAttribute("target");
    alert(target); // Outputs: _blank
</script>

ex:
---
<button type="button" id="myBtn">Click Me</button>

<script>
    // Selecting the element
    var btn = document.getElementById("myBtn");

    // Setting new attributes
    btn.setAttribute("class", "click-btn");
    btn.setAttribute("disabled", "");
</script>

ex:
---

<a href="#" id="myLink">Tutorial Republic</a>

<script>
    // Selecting the element
    var link = document.getElementById("myLink");

    // Changing the href attribute value
    link.setAttribute("href", "https://www.tutorialrepublic.com");
</script>

ex:
---

<a href="https://www.google.com/" id="myLink">Google</a>

<script>
```

```
    // Selecting the element
    var link = document.getElementById("myLink");

    // Removing the href attribute
    link.removeAttribute("href");
</script>

JavaScript DOM Manipulation:
-------------------------
ex:
---
<div id="main">
    <h1 id="title">Hello World!</h1>
    <p id="hint">This is a simple paragraph.</p>
</div>

<script>
// Creating a new div element
var newDiv = document.createElement("div");

// Creating a text node
var newContent = document.createTextNode("Hi, how are you doing?");

// Adding the text node to the newly created div
newDiv.appendChild(newContent);

// Adding the newly created element and its content into the DOM
var currentDiv = document.getElementById("main");
document.body.appendChild(newDiv, currentDiv);
</script>

ex:
---
<div id="main">
    <h1 id="title">Hello World!</h1>
    <p id="hint">This is a simple paragraph.</p>
</div>

<script>
// Creating a new div element
var newDiv = document.createElement("div");

// Creating a text node
var newContent = document.createTextNode("Hi, how are you doing?");

// Adding the text node to the newly created div
newDiv.appendChild(newContent);

// Adding the newly created element and its content into the DOM
var currentDiv = document.getElementById("main");
```

```
document.body.insertBefore(newDiv, currentDiv);
</script>

ex:
---
<div id="main">
    <h1 id="title">Hello World!</h1>
    <p id="hint">This is a simple paragraph.</p>
</div>

<script>
// Getting inner HTML conents
var contents = document.getElementById("main").innerHTML;
alert(contents); // Outputs inner html contents

// Setting inner HTML contents
var mainDiv = document.getElementById("main");
mainDiv.innerHTML = "<p>This is <em>newly inserted</em> paragraph.</p>";
</script>
ex:
---
<!-- beforebegin -->
<div id="main">
    <!-- afterbegin -->
    <h1 id="title">Hello World!</h1>
    <!-- beforeend -->
</div>
<!-- afterend -->

<script>
// Selecting target element
var mainDiv = document.getElementById("main");

// Inserting HTML just before the element itself, as a previous sibling
mainDiv.insertAdjacentHTML('beforebegin', '<p>This is paragraph one.</p>');

// Inserting HTML just inside the element, before its first child
mainDiv.insertAdjacentHTML('afterbegin', '<p>This is paragraph two.</p>');

// Inserting HTML just inside the element, after its last child
mainDiv.insertAdjacentHTML('beforeend', '<p>This is paragraph three.</p>');

// Inserting HTML just after the element itself, as a next sibling
mainDiv.insertAdjacentHTML('afterend', '<p>This is paragraph four.</p>');
</script>

ex:
-
<div id="main">
    <h1 id="title">Hello World!</h1>
    <p id="hint">This is a simple paragraph.</p>
</div>

<script>
var parentElem = document.getElementById("main");
var childElem = document.getElementById("hint");
parentElem.removeChild(childElem);
</script>

ex:
---
<div id="main">
    <h1 id="title">Hello World!</h1>
    <p id="hint">This is a simple paragraph.</p>
</div>

<script>
var childElem = document.getElementById("hint");
childElem.parentNode.removeChild(childElem);
</script>

ex:
---
<div id="main">
    <h1 id="title">Hello World!</h1>
    <p id="hint">This is a simple paragraph.</p>
</div>

<script>
var parentElem = document.getElementById("main");
var oldPara = document.getElementById("hint");

// Creating new elememt
var newPara = document.createElement("p");
var newContent = document.createTextNode("This is a new paragraph.");
newPara.appendChild(newContent);

// Replacing old paragraph with newly created paragraph
parentElem.replaceChild(newPara, oldPara);
</script>

BOM:
-------------------------->>>>>

Calculating Width and Height of the Window:
--------------

EX:
---
<script>
```

```
function windowSize(){
    var w = window.innerWidth;
    var h = window.innerHeight;
    alert("Width: " + w + ", " + "Height: " + h);
}
</script>

<button type="button" onclick="windowSize();">Get Window Size</button>

EX:
--
<script>
function windowSize(){
    var w = document.documentElement.clientWidth;
    var h = document.documentElement.clientHeight;
    alert("Width: " + w + ", " + "Height: " + h);
}
</script>

<button type="button" onclick="windowSize();">Get Window Size</button>

JavaScript Window Screen:
------------------------

EX:
--
<script>
function getResolution() {
    alert("Your screen is: " + screen.width + "x" + screen.height);
}
</script>

<button type="button" onclick="getResolution();">Get Resolution</button>

EX:
---
<script>
function getAvailSize() {
    alert("Available Screen Width: " + screen.availWidth + ", Height: " +
screen.availHeight);
}
</script>

<button type="button" onclick="getAvailSize();">Get Available Size</button>
EX:
---
<script>
function getColorDepth() {
    alert("Your screen color depth is: " + screen.colorDepth);
}
```

```
</script>

<button type="button" onclick="getColorDepth();">Get Color Depth</button>

EX:
---
<script>
function getPixelDepth() {
    alert("Your screen pixel depth is: " + screen.pixelDepth);
}
</script>

<button type="button" onclick="getPixelDepth();">Get Pixel Depth</button>

JavaScript Window Location:
===============================
EX:
---
<script>
function getURL() {
    alert("The URL of this page is: " + window.location.href);
}
</script>

<button type="button" onclick="getURL();">Get Page URL</button>

EX:
---

// Prints complete URL
document.write(window.location.href);

// Prints protocol like http: or https:
document.write(window.location.protocol);

// Prints hostname with port like localhost or localhost:3000
document.write(window.location.host);

// Prints hostname like localhost or www.example.com
document.write(window.location.hostname);

// Prints port number like 3000
document.write(window.location.port);

// Prints pathname like /products/search.php
document.write(window.location.pathname);

// Prints query string like ?q=ipad
document.write(window.location.search);
```

```
// Prints fragment identifier like #featured
document.write(window.location.hash);

EX:
---
<script>
function loadHomePage() {
    window.location.assign("https://www.tutorialrepublic.com");
}
</script>

<button type="button" onclick="loadHomePage();">Load Home Page</button>

EX:
---
<script>
function loadHomePage(){
    window.location.replace("https://www.tutorialrepublic.com");
}
</script>

<button type="button" onclick="loadHomePage();">Load Home Page</button>

EX:
---
<script>
function loadHomePage() {
    window.location.href = "https://www.tutorialrepublic.com";
}
</script>

<button type="button" onclick="loadHomePage();">Load Home Page</button>

EX:
---
<script>
function forceReload() {
    window.location.reload(true);
}
</script>

<button type="button" onclick="forceReload();">Reload Page</button>

JavaScript Window History:
=========================

EX:
--
<script>
function getViews() {
    alert("You've accessed " + history.length + " web pages in this session.");
}
</script>

<button type="button" onclick="getViews();">Get Views Count</button>

EX:
--
<script>
function goBack() {
    window.history.back();
}
</script>

<button type="button" onclick="goBack();">Go Back</button>
EX:
----
<script>
function goForward() {
    window.history.forward();
}
</script>

<button type="button" onclick="goForward();">Go Forward</button>

EX:
--window.history.go(-2);  // Go back two pages
window.history.go(-1); // Go back one page
window.history.go(0);  // Reload the current page
window.history.go(1);  // Go forward one page
window.history.go(2);  // Go forward two pages

JavaScript Window Navigator:
------------------------------
EX:
--
<script>
function checkConnectionStatus() {
    if(navigator.onLine) {
        alert("Application is online.");
    } else {
        alert("Application is offline.");
    }
}
</script>

<button type="button" onclick="checkConnectionStatus();">Check Connection
Status</button>
EX:
---
```

```
<script>
function goOnline() {
    // Action to be performed when your application goes online
    alert("And we're back!");
}

function goOffline() {
    // Action to be performed when your application goes offline
    alert("Hey, it looks like you're offline.");
}

// Attaching event handler for the online event
window.addEventListener("online", goOnline);

// Attaching event handler for the offline event
window.addEventListener("offline", goOffline);
</script>

<p>Toggle your internet connection on/off to see how it works.</p>
EX:
---
<script>
function checkCookieEnabled() {
    if(navigator.cookieEnabled) {
        alert("Cookies are enabled in your browser.");
    } else {
        alert("Cookies are disabled in your browser.");
    }
}
</script>

<button type="button" onclick="checkCookieEnabled();">Check If Cookies are
Enabled</button>

EX:
====
<script>
function checkLanguage() {
    alert("Your browser's UI language is: " + navigator.language);
}
</script>

<button type="button" onclick="checkLanguage();">Check Language</button>

EX:
---
<script>
function getBrowserInformation() {
        var info = "\n App Name: " + navigator.appName;
           info += "\n App Version: " + navigator.appVersion;
           info += "\n App Code Name: " + navigator.appCodeName;
           info += "\n User Agent: " + navigator.userAgent;
           info += "\n Platform: " + navigator.platform;

    alert("Here're the information related to your browser: " + info);
}
</script>

<button type="button" onclick="getBrowserInformation();">Get Browser
Information</button>
EX:
---
<script>
function checkJavaEnabled() {
    if(navigator.javaEnabled()) {
        alert("Your browser is Java enabled.");
    } else {
        alert("Your browser is not Java enabled.");
    }
}
</script>

<button type="button" onclick="checkJavaEnabled();">Check If Java is
Enabled</button>
```