

# Document Object Model(DOM) & Browser Object Model(BOM)

## DOM & BOM IN JS:

### Window Vs Document:

Window is the main container or we can say the global object and any operations related to entire browser window can be a part of window object. All the members like objects, methods, or properties if they are the part of window object then we don't refer the window object.(window, window.location, do in browser console) window has methods, properties and object. Ex: setTimeout()or setInterval() are the methods.

Window has methods, properties, and object. Ex: ex: setTimeout()or setInterval() are the methods. Where as document is the object of the window and it also has a screen object with properties describing the physical display. We have seen the methods and object of the global object that is window. But what about the properties of the window objects. Innerheight, innerwidth and many more properties of window object

Document is the DOM is the child of window object. Where in the DOM we need to refer the document. If we want to use the document object, method or properties. Where as document is the object of the window and it also has a screen object with properties describing the physical display. Document is just the object of the global object that is window. Which deals with the document the html elements themselves. In DOM all HTML elements are defined as objects so it will have both property and methods. The document object represents your web page. If you want to access any element in an HTML page you always start with accessing the document object.

### DOM VS BOM:

Document Object Model(DOM) is an Application Programming Interface(API) for manipulating HTML and XML documents.(add, remove, and modify parts of the document/HTML). When a web page is loaded the browser creates a "Document Object Model" of the page. In DOM tree the DOCUMENT is the root node or Object.

Browser Object Model(BOM), which deals with browser components aside from the document, like history, location, navigator, and screen (as well as some others that vary by browser) or in simple meaning all the window operations which comes under BOM are performed using BOM.

The window object represents a window in browser. An object of window is created automatically by the browser. All global javascript objects, functions and variables with the var keyword automatically become members of the window objects. Global variables are properties of the window object. Global functions are methods of the window object. If you use let and const it will show you undefined for this concept.

## Navigate Through The DOM:

1. In DOM all HTML elements are defined as objects. So it will have both property and method. The document object represents your web page.
2. If you want to access any elements in an HTML page, you always start with accessing the document object.

With the Object Model javascript gets all the power it needs to create dynamic HTML

1. JS CAN CHANGE ALL THE HTML ELEMENTS IN THE PAGE
2. JS CAN CHANGE ALL THE HTML ATTRIBUTES IN THE PAGE
3. JS CAN CHANGE ALL THE CSS STYLE IN THE PAGE
4. JS CAN REMOVE EXISTING HTML ELEMENTS AND ATTRIBUTES
5. JS CAN ADD NEW HTML ELEMENTS AND ATTRIBUTES
6. JS CAN REACT TO ALL EXISTING HTML EVENTS IN THE PAGE
7. JS CAN CREATE NEW HTML EVENTS IN THE PAGE

## Select An Element By ID(#):

```
// /*SELECT AN ELEMENT BY ID*/  
// let oneel = document.getElementById("one");  
// console.log(oneel.innerHTML);  
// console.log("INPUT VALUE = "+oneel);  
  
// let twoel = document.getElementById("slt");  
// console.log(twoel.innerHTML+"Atish");  
// twoel.innerHTML = "<h2>ATISH</h2>";  
// console.log("value = "+twoel);
```

```
<!-- SELECT AN ELEMENT BY ID -->  
<input type="text" id="one">  
<ul>  
  <li>list1</li>  
  <li id="slt">list2</li>  
  <li>list3</li>  
</ul>
```

## Select Element By Class(.):

```
// /*SELECT AN ELEMENT BY CLASS*/  
// let cslt = document.getElementsByClassName("slt");  
// console.log(cslt);  
// console.log(cslt.length);  
// for(let i = 0; i < cslt.length; i++)  
// {  
//   cslt[i].innerHTML="result";  
// }
```

```
<!-- SELECT AN ELEMENT BY CLASS -->  
<ul>  
  <li class="slt">list1</li>  
  <li class="slt">list2</li>  
  <li class="slt">list3</li>  
</ul>
```

## Select Element By Tag Name:

```
<!-- SELECT AN ELEMENT BY TAG NAME -->
<ul>
  <li>LIST</li>
  <li>LIST</li>
  <li>LIST</li>
</ul>

<h5>heading5</h5>
<h5>heading5</h5>
<h5>heading5</h5>
```

```
// /*SELECT AN ELEMENT BY TAG NAME*/
// let cng = document.getElementsByTagName("h5");
// for(let i = 0; i < cng.length; i++)
// {
//   cng[i].innerHTML="result";
// }
```

## Select Element By Query Selector:

```
// /*SELECT AN ELEMENT BY QUERY SELECTOR*/
// let qs = document.querySelectorAll("p.intro");
// console.log(qs);
// for(let i = 0; i < cng.length; i++)
// {
//   qs[i].innerHTML="QUERY SELECTOR";
// }
```

```
<!-- SELECT AN ELEMENT BY QUERY SELECTOR -->
<p class="intro">PARAGRAPH1</p>
<p>PARAGRAPH2</p>
<p class="intro">PARAGRAPH3</p>
<p class="intro">PARAGRAPH4</p>
```

## Sample test:

```
// /*Test*/
// let test1 = document.getElementById("chk");
// let test2 = test1.getElementsByTagName("p");
// for(let i = 0; i < test2.length; i++)
// {
//   test2[i].innerHTML="DIV TO P SELECTION";
// }
```

```
<div id="chk">
  <p>Paragraph1</p>
  <p>Paragraph2</p>
</div>
```

## Traverse Element:

```
<ul id="tra">
  <li>Add1</li>
  <li>Add2</li>
  <li>Add3</li>
  <li>Add4</li>
  <li>Add5</li>
</ul>
```

```
let ul1 = document.getElementById("tra");
let li1 = ul1.getElementsByTagName("li");
for(let i = 0; i < li1.length; i++)
{
  li1[i].textContent = "Element" + i;
}
```

```
<!-- TRAVERSING ELEMENTS IN JS -->
```

```
<ul>
```

```
  <li id="li1">list1</li>
```

```
  <li>list2</li>
```

```
  <li>list3</li>
```

```
  <li>list4</li>
```

```
</ul>
```

```
<ul id="ul1">
```

```
  <li>list a</li>
```

```
  <li id="tst1">list b</li>
```

```
  <li>list c</li>
```

```
  <li>list d</li>
```

```
</ul>
```

```
// /*TRAVERSING ELEMENTS IN JS*/  
// let lst = document.getElementById("li1");  
// console.log(lst);  
// let prnt = lst.parentElement;  
// console.log(prnt);  
// prnt.innerHTML = "hello";
```

```
// let ust = document.getElementById("ul1");  
// console.log(ust);  
// let lli = ust.firstElementChild;  
// console.log(lli);  
// let lli1 = ust.lastElementChild;  
// console.log(lli1);  
// let lia = ust.children;  
// console.log(lia);
```

```
// let test3 = document.getElementById("tst1");  
// console.log(test3);  
// let test4 = test3.previousElementSibling;  
// console.log(test4);  
// let test5 = test3.nextElementSibling;  
// console.log(test5);
```

Change HTML(Inner HTML) in JS fetch HTML element & change HTML element:

```
// /*CHANGE HTML(INNERHTML) IN JS FETCH HTML ELEMENT AND CHANGE  
HTML ELEMENT*/
```

```
// let itr = document.getElementById("intro");
```

```
// console.log(itr);
```

```
// let itr1 = itr.innerHTML;
```

```
// console.log(itr1);
```

```
// itr.innerHTML = "<h2>This Is A Heading</h2>";
```

```
<!-- CHANGE HTML(INNERHTML) IN JS -->
```

```
<div id="intro">
```

```
  <p>THIS IS A PARAGRAPH</p>
```

```
</div>
```

## Create & Append Element in js:

```
<!-- CREATE AND APPEND ELEMENT IN JS -->
```

```
<div id="apnd">  
  <p id="p1">THIS IS A PARAGRAPH</p>  
</div>
```

```
<ul id="ulsl">  
  <li>Atish</li>  
  <li>Lipun</li>  
  <li>Mritunjay</li>  
  <li>Satish</li>  
</ul>
```

```
// /*CREATE AND APPEND ELEMENT IN JS*/  
// let apd1 = document.getElementById("apnd");  
  
// let apd2 = document.createElement("h2");  
// apd2.className = "clsh2 hi";  
// apd2.id = "ABC1"  
  
// let apd3 = document.createTextNode("This is heading2"); //using  
method  
// apd2.appendChild(apd3);  
  
// //apd1.appendChild(apd2);  
  
// document.body.appendChild(apd2);  
  
// apd2.textContent = "ATISH KUMAR SAHU"; //using property  
  
// let para1 = document.getElementById("p1");  
// console.log(para1.textContent);  
  
// let uls = document.getElementById("ulsl");  
// let item = document.createElement("li");  
// item.textContent = "New List5";  
// uls.appendChild(item);
```

## Insert Before In JS:

```
<!-- INSERT BEFORE IN JS -->
```

```
<ul id="IB">  
  <li>ATISH 1</li>  
  <li>ATISH 2</li>  
  <li>ATISH 3</li>  
  <li>ATISH 4</li>  
</ul>
```

```
// /*INSERT BEFORE IN JS*/
```

```
// let ulib = document.getElementById("IB");  
// let ulib1 = document.createElement("li");  
// ulib1.textContent="NEW ATISH";
```

```
// let pos = ulib.firstChild;  
// ulib.insertBefore(ulib1,pos);
```

```
// let ns = pos.nextSibling;  
// ulib.insertBefore(ulib1,ns);
```

```
// let pr = document.body;  
// let elm1 = document.createElement("h2");  
// elm1.textContent = "Heading2 ATISH";
```

```
// let lst1 = document.getElementById("IB");
```

```
// pr.insertBefore(elm1,lst1);
```

## Remove child elements In JS:

```
// /*REMOVE CHILD ELEMENT IN JS*/  
// let ulmn = document.getElementById("menu");  
// let eie = ulmn.firstChild;  
// ulmn.removeChild(eie);  
  
// let eie1 = ulmn.firstChild.nextElementSibling;  
// ulmn.removeChild(eie1);  
  
// document.body.removeChild(ulmn);
```

```
<!-- REMOVE CHILD ELEMENT IN JS -->
```

```
<ul id="menu">  
  <li>HOME</li>  
  <li>PRODUCT</li>  
  <li>ABOUT</li>  
</ul>
```

## Clone Element In JS:

```
// /*CLONE ELEMENT*/  
// let clo = document.getElementById("cln");  
// let clo1 = clo.cloneNode(true);  
// clo1.id="cln1";  
// console.log(clo1);  
// let clo2 = clo.cloneNode();  
// console.log(clo2);  
// document.body.appendChild(clo1);
```

```
<!-- CLONE ELEMENT -->
```

```
<ul id="cln">  
  <li>LIST 01</li>  
  <li>LIST 02</li>  
  <li>LIST 03</li>  
  <li>LIST 04</li>  
  <li>LIST 05</li>  
</ul>
```

## Replace Element:

```
<!-- REPLACE ELEMENT -->
<ul id="mnu">
  <li>MENU1</li>
  <li>MENU2</li>
  <li>MENU3</li>
</ul>
```

```
// /*REPLACE ELEMENT*/
// let mn1 = document.getElementById("mnu");
// let mn2 = document.createElement("li");
// mn2.textContent = "Services";
// let replace = mn1.firstElementChild.nextElementSibling;
// mn1.replaceChild(mn2,replace);
```

## Insert Adjacent Element:

```
// /*INSERT ADJACENTHTML*/
// let ia1 = document.getElementById("ia");

// let html1 = "<h1>BEFORE BEGIN</H1>";
// ia1.insertAdjacentHTML('beforebegin',html1);

// let html2 = "<h1>AFTER BEGIN</h1>";
// ia1.insertAdjacentHTML('afterbegin',html2);

// let html3 = "<h1>BEFORE END</h1>";
// ia1.insertAdjacentHTML('beforeend',html3);

// let html4 = "<h1>AFTER END</h1>";
// ia1.insertAdjacentHTML('afterend',html4);
```

```
<!-- INSERT ADJACENTHTML -->

<!-- beforebegin -->
<div id="ia">
  <!-- afterbegin -->
  <h2>HEADING2</h2>
  <p>THIS IS A P TAG.</p>
  <!-- beforeend -->
</div>
<!-- afterend -->
```

## Change Attribute & Inline Style Through JS:

```
// /*CHANGE ATTRIBUTE IN JAVASCRIPT*/

// let btn1 = document.getElementById("btn");
// btn1.setAttribute("name","form1");

// let nm1 = btn1.getAttribute("name");
// console.log(nm1);

// let rva = btn1.removeAttribute("name");

// let ha = btn1.hasAttribute("name");
// console.log(ha);

// /*INLINE STYLE*/
// btn1.style.cssText = "background-color: red; color: white;";
// replace the css
// btn1.setAttribute("style","width: 100px; height: 200px");
// replace the css
// btn1.style.color = "blue";
```

```
<!-- CHANGE ATTRIBUTE IN JAVASCRIPT --><!-- INLINE STYLE -->

<button id="btn">send</button>
```

## Get Computed CSS:

```
// /*GET COMPUTED CSS */  
// let cc1 = document.getElementById("btncc");  
// let cc2 = getComputedStyle(cc1);  
// console.log(cc2);  
// console.log(cc2.color);
```

```
<!-- GET COMPUTED CSS -->
```

```
<button id="btncc">send</button>
```

## Change CSS class & Width & Height of element through JS:

```
<!-- CHANGE CSS CLASSES --><!-- WIDTH AND HEIGHT OF AN ELEMENT -->
```

```
<div id="box" class="bg">  
  <p>THIS IS A DUMY TEXT.</p>  
</div>
```

```
// /*CHANGE CSS CLASSES*/  
// let ccc1 = document.getElementById("box");  
// console.log(ccc1.className);  
// ccc1.className += " dim";  
// console.log(ccc1.classList);  
  
// for(let css of ccc1.classList){  
  
//   console.log(css);  
// }  
// ccc1.classList.add("dim");  
// ccc1.classList.remove("dim");  
// ccc1.classList.replace("bg","dim");  
// let ccc2 = ccc1.classList.contains("bg");  
// console.log(ccc2);  
// ccc1.classList.toggle("bg"); //if available then take out or if  
// not available then add it.  
  
// /*WIDTH AND HEIGHT OF AN ELEMENT*/  
  
// let wh1 = document.getElementById("box");  
  
// let wh2 = wh1.offsetWidth; //with border  
// let wh3 = wh1.offsetHeight; //with border  
// console.log(`${wh2+" "+wh3}`);  
  
// let wh4 = wh1.clientWidth; //without border  
// let wh5 = wh1.clientHeight; //without border  
// console.log(`${wh4+" "+wh5}`);
```



## Document Object Model Events:

```
<!-- DOCUMENT OBJECT MODEL(DOM) EVENTS -->
<button id="btne" onclick="btnclk()">click me!</button>

<button id="btne1">CLICK HERE</button>
```

```
// /*DOCUMENT OBJECT MODEL(DOM) EVENTS*/
// function btnclk()
// {
//   alert("button was clicked.");
// }
// function btnclk1(){
//   alert("YOU CLICKED");
// }
// let btne = document.getElementById("btne1");
// btne.addEventListener('click',btnclk1);
// btne.addEventListener('click', function(){
//   alert("BUTTON WAS ONCLICKED");
// });
// btne.addEventListener('mouseover',function(){
//   console.log("YOU HOVER MOUSE ON THE BUTTON");
// });
// btne.addEventListener('mouseout',function(){
//   console.log("YOU OUT MOUSE ON THE BUTTON");
// });
```

## Remove eventlistener:

```
<!-- REMOVE EVENTLISTENER -->

<button id="rebtn">REMOVE BUTTON</button>
```

```
// /*REMOVE EVENTLISTENER*/
// let btn = document.getElementById("rebtn");
// function remove(){
//   console.log("remove1 action activated.")
// }
// function remove1(){
//   console.log("remove2 action activated.")
// }
// btn.addEventListener('click', remove);
// btn.addEventListener('click', remove1);
// btn.removeEventListener('click', remove1);
```

## Page Load Events:

### DOM Content Loaded:

The browser fully loaded HTML and completed building the DOM tree. However, it isn't loaded external resources like stylesheets and images. In this even you can start selecting DOM nodes or initialize the interface.

### LOAD:

The browser fully loaded the html and also external resources like images and stylesheets. When you leave the page, the following events are in sequence.

```
// window.addEventListener('DOMContentLoaded',function(){
//   console.log("DOM TREE CREATED.");
// });
// window.addEventListener('load',function(){
//   console.log("FULLY LOADED.");
// });
```

### Mouse Events In JS:

```
<!-- MOUSE EVENTS IN JS -->
<div id="mbx" onclick="ME()"></div>
<div id="mbx1" oncontextmenu="ME1()"></div>
<div id="mbx2" ondblclick="ME2()"></div>
<div id="mbx3" onmousedown="ME3()"></div>
<div id="mbx4" onmouseup="ME4()"></div>
<div id="mbx5" onmouseover="ME5()"></div>
<div id="mbx6" onmouseout="ME6()"></div>
```

```
// /*MOUSE EVENTS IN JS*/
// function ME(){
//   alert("ON CLICK EVENT");
// }
```

```
// function ME1(){
//   alert("YOU RIGHT CLICKED FROM MOUSE");
// }
```

```
// function ME2(){
//   alert("YOU DOUBLE CLICKED FROM MOUSE");
// }
```


```
// function ME3(){
//   alert("YOU ACTIVATED THE MOUSEDOWN EVENT");
// }
```

```
// function ME4(){
//   alert("YOU ACTIVATED THE MOUSEUP EVENT");
// }
```

```
// function ME5(){
//   alert("YOU ACTIVATED THE MOUSEOVER EVENT");
// }
```

```
// function ME6(){
//   alert("YOU ACTIVATED THE MOUSEOUT EVENT");
// }
```

## Key Down Event:

```
<!-- KEY DOWN EVENT -->
<div id="kbox1" style="height: 100px; width: 100px;
background-color: red;">Key Down Event</div>
```

```
// /*KEY DOWN EVENT*/
// window.addEventListener('keydown',checkkey);
// function checkkey(kchk1){ //key down means key press and hold
//   console.log(kchk1.key);
// }

// window.addEventListener('keyup',checkkey1);
// function checkkey1(kchk2){ //keyup means after key press and hold
//   console.log(kchk2.key);
// }
```

## Scroll Event In JS:

```
// /*SCROLL EVENTS*/
// window.addEventListener('scroll',function(){
//   console.log("WINDOW SCROLLING");
// });

// window.addEventListener('wheel',function(we){
//   if(we.deltaY < 0)
//   {
//     console.log("SCROLLING UP");
//   }
//   else if(we.deltaY > 0){
//     console.log("SCROLLING DOWN");
//   }
// });

// window.addEventListener('scroll',function(){
//   if(window.pageYOffset > 800){
//     this.document.body.style.background = "blue";
//   }
// });
```

## Event On Form In JS:

```
<!-- EVENTS ON FORM IN JS -->
```

```
<form action="">
|   <input type="text" id="l1">
</form>
```

```
// /*EVENTS ON FORM IN JS*/
// let ip1 = document.getElementById("l1");
// ip1.addEventListener('focus',mfocus);
// ip1.addEventListener('blur',mblur);
// ip1.addEventListener('change', function(){
//   console.log(this.value);
// });
// ip1.addEventListener('input',function(){
//   console.log(this.value);
// })

// function mfocus()
// {
//   ip1.style.background = "yellow";
// }
// function mblur()
// {
//   ip1.style.background = "white";
// }
```

## Event Bubbling & Event Capturing:

### Event Bubbling:

In the event bubbling model and event starts at the most specific element and then flows upward toward the least specific element(the document or even window.) When you click button the click the button, the click event occurs in the following order. Ex: button, div with the id container, body, html, document

### Event Capturing:

In the event capturing model an event starts at the least specific element and flows downward toward the most specific element. When you click the button, the click event occurs in the following order. Ex: document, html, body, div with id container, button

### Syntax:

addEventListener(event, function, usecapture);

The above syntax the default value is false, which will use the bubbling propagation. When the value is set to true, the event uses the capturing propagation.

```

<!-- Event Bubbling & Event Capturing -->
<div id="EB">
  <button id="bt63">
    click me!
  </button>
</div>

```

```

// let d = document.getElementById("EB");
// let b = document.getElementById("bt63");
// //if you did not give true in third parameter it means default
// value is false.default false means bubbling. true means capturing
// b.addEventListener('click', btnClicked);
// d.addEventListener('click', divClicked);
// document.body.addEventListener('click', bodyClicked);

// function btnClicked(event){
//   console.log("Button Clicked");
//   event.stopPropagation();
//   /*stopPropagation means it will stop the operation of
// bubbling and capturing of the particular element.*/
// }
// function divClicked(){
//   console.log("Div Clicked");
// }
// function bodyClicked(){
//   console.log("Body Clicked");
// }

```

## Prevent Default:

```

// /*Prevent Default */
// let link = document.getElementById("YT");
// link.addEventListener('click', function(e){
//   console.log("Link Clicked!");
//   e.preventDefault();
// });

// let form = document.getElementById("frm64");
// form.addEventListener('submit', function(e){
//   e.preventDefault();
// })

```

```

<!-- Prevent Default -->
<a href="http://www.youtube.com" id="YT">YouTube Click!</a>

<form action="" id="frm64">
  <input type="text">
  <input type="submit">
</form>

```

## BOM Window:

```
// /*Window */
// console.log(window.innerHeight);
// console.log(window.innerWidth);
// console.log(window.outerHeight);
// console.log(window.outerWidth);

// let wb1 = document.getElementById("wbtn");
// let url = "https://www.google.com/";
// let naem = 'Atish';
// let win1;
// let features = "height = 500, width = 400";
// wb1.addEventListener('click', function(){
//     win1 = window.open(url, naem, features);
// })

// let wb2 = document.getElementById("wbtn1");
// wb2.addEventListener('click', function(){
//     window.open("https://www.yahoo.com", naem, features);
// })

// let wb3 = document.getElementById("wbtn2");
// wb3.addEventListener('click', function(){
//     win1.close();
// })
```

```
<!-- Window -->
<button id="wbtn">
|   google
</button>

<button id="wbtn1">
|   Yahoo!
</button>

<button id="wbtn2">
|   close!
</button>
```

## Time Out & Time Interval:

```
<!-- Time Out & Time Interval -->
<button id="TOTI">
  click click!
</button>
```

```
// /*Time Out & Time Interval */
// let timeout = setTimeout(myfunction, 5000);
// function myfunction(){
//   alert("please subscribe");
// }
// clearTimeout(timeout);

// let timeinterval = setInterval(myfunction1, 3000);
// function myfunction1(){
//   console.log("I Love You");
// }

// let btt1 = document.getElementById("TOTI");
// btt1.addEventListener('click',function(){
//   clearInterval(timeinterval);
// })
```

## Location Object:

```
// /*Loations Object */
// console.log(location.href);
// console.log(location.pathname);
// console.log(location.protocol);
// console.log(location.port);

// function lof()
// {
//   // window.location = "https://www.google.com";
//   // location.href = "https://www.youtube.com";
//   location.assign("https://microsoft.com");

//   /*above 3 are same things */

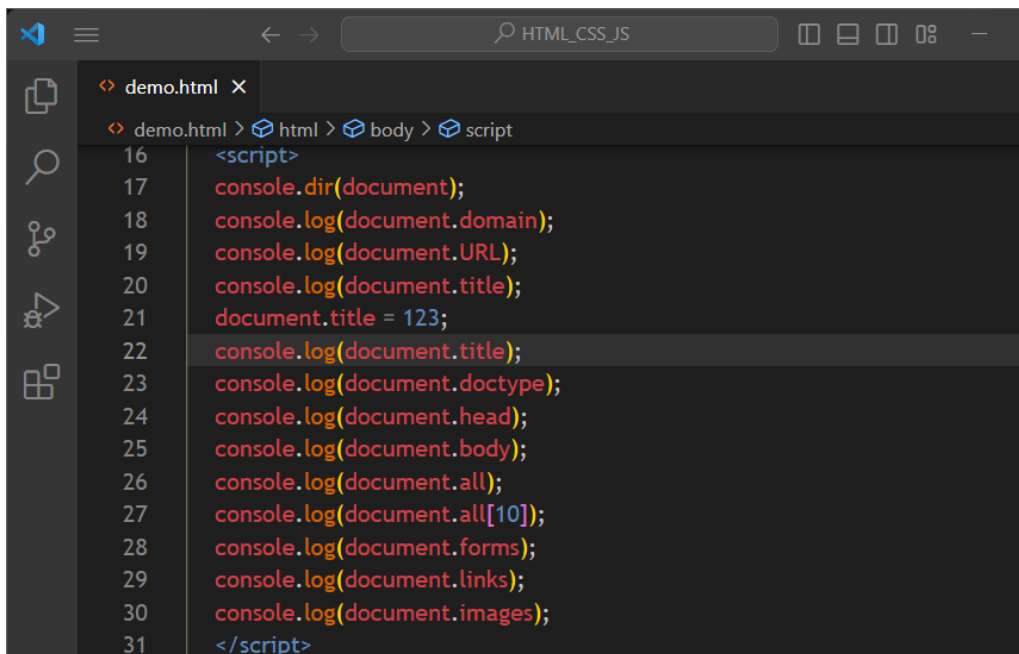
//   //location.replace("https://www.google.com"); //no back button
//   location.reload(); //refresh the page.
// }
```

```
<!-- Location Object -->
<button id="LO" onclick="lof()">MEWO </button>
```

## Navigator Object & Screen Object:

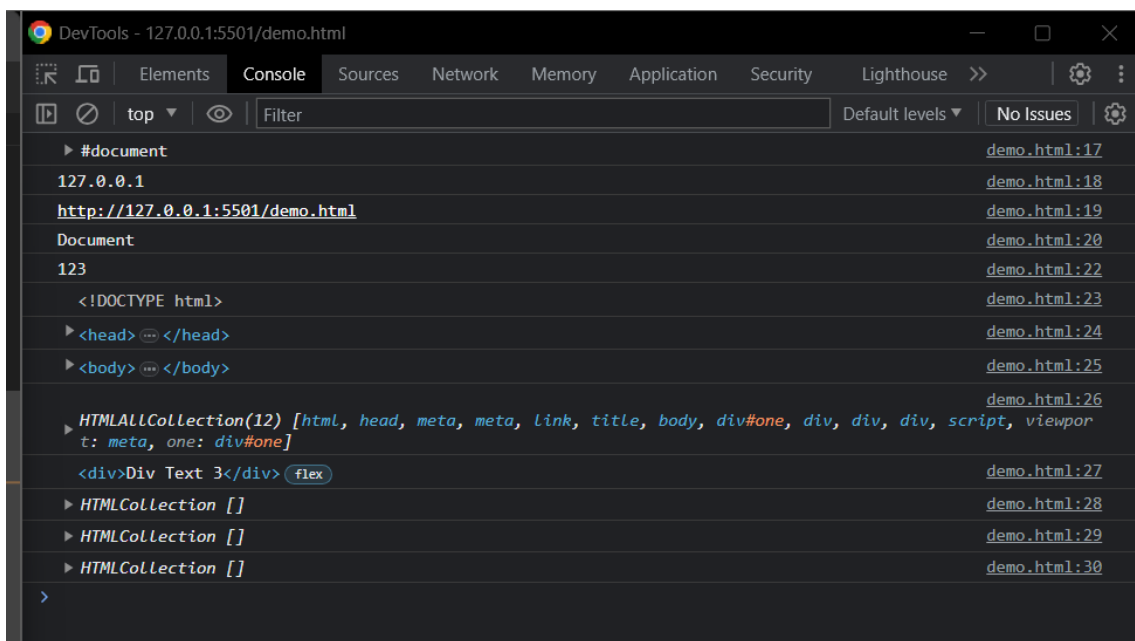
```
// /*Navigator Object */
// console.log(navigator.appName);
// console.log(navigator.appVersion);
// console.log(navigator.appCodeName);
// console.log(navigator.cookieEnabled);
// console.log(navigator.userAgent);
// console.log(navigator.platform);
// console.log(navigator.javaEnabled());

// /*Screen Object */
// console.log(screen.width);
// console.log(screen.height);
// console.log(screen.colorDepth);
// console.log(screen.orientation);
```



The screenshot shows a code editor with a file named 'demo.html'. The script block contains the following code:

```
<script>
16 console.dir(document);
17 console.log(document.domain);
18 console.log(document.URL);
19 console.log(document.title);
20 document.title = 123;
21 console.log(document.title);
22 console.log(document.doctype);
23 console.log(document.head);
24 console.log(document.body);
25 console.log(document.all);
26 console.log(document.all[10]);
27 console.log(document.forms);
28 console.log(document.links);
29 console.log(document.images);
30
31 </script>
```



The screenshot shows the Chrome DevTools Console with the 'Console' tab selected. The output of the JavaScript code is displayed as follows:

- #document (demo.html:17)
- 127.0.0.1 (demo.html:18)
- http://127.0.0.1:5501/demo.html (demo.html:19)
- Document (demo.html:20)
- 123 (demo.html:22)
- <!DOCTYPE html> (demo.html:23)
- <head> (demo.html:24)
- <body> (demo.html:25)
- HTMLAllCollection(12) [html, head, meta, meta, link, title, body, div#one, div, div, div, script, viewport] (demo.html:26)
- <div>Div Text 3</div> (demo.html:27)
- HTMLCollection [] (demo.html:28)
- HTMLCollection [] (demo.html:29)
- HTMLCollection [] (demo.html:30)



## 48. Module In javascript:

```
JS demo.js
1 // const prompt = require("prompt-sync")();s
2 // import {name, withdraw} from './modulejs/one.js';
3 // import {code} from './modulejs/one.js';
4 //import {age} from './modulejs/one.js'; //error
5 // console.log(name);
6 // code();
7 //console.log(age); //error
8 // withdraw();
9
10 // import {withdraw as WD, depos} from './modulejs/
two.js';
11 // WD();
12 // depos();
13
14 // import * as on from './modulejs/one.js';
15 // on.code();
16
17 // import {default as def} from './modulejs/one.
js';
18 // def();
19
20 // import {code} from './modulejs/one.js';
21 // code();
22
23
24
25
26
27
28
29
30
```

```
modulejs > JS one.js > default
1 import {withdraw as wwd} from './two.js';
2 export default function(){
3   console.log("Default Operation From
Default Function One.js");
4 }
5 export let name = "Atish Kumar Sahu";
6 let age = 15;
7 export function code(){
8   console.log("Coding Operation From Code
Function One.js");
9   wwd();
10 }
11 export function withdraw(){
12   console.log("Withdraw Operation From
Withdraw Function One.js");
13 }
14 function cook(){
15   console.log("Cooking Operation From Cook
Function One.js");
16 }
17 /*
18 use export keyword to give permission to the
other files to acces only those variables,
function, object, class, array where the
export keyword is used in starting of them.
19
20 in a single js file you can add only one
default fuction which has no name like it is
a anonymous function. and infront of default
function you must have to add export keyword
and default keyword.
21 */
```

```
modulejs > JS two.js > ...
1 export default function(){
2   console.log("Default
Operation From Default
Function Two.js");
3 }
4
5 let accnum = 19829588299865;
6 let acctype = "Software
Developer";
7
8 function withdraw(){
9   console.log("Withdraw
Operation From Withdraw
Function Two.js");
10 }
11 function depos(){
12   console.log("Deposite
Operation from Depos
Function Two.js");
13 }
14
15 export {withdraw, depos};
```

```
JS demo.js x ...
JS demo.js
43 // import {circ as c } from './
   modules1/shapes/circle.js';
44
45 // import {trang as t } from './
   modules1/shapes/traingle.js';
46
47 // import {sqr as s } from './
   modules1/shapes/square.js';
48
49 // c(); t(); s();
50
51 //import{ c, t, s} from './
   modules1/shape.js';
52
53 //c(); t(); s();

JS shape.js x ...
modules1 > JS shape.js
1 export {circ as c } from './
   shapes/circle.js';
2 export {trang as t } from './
   shapes/traingle.js';
3 export {sqr as s } from './
   shapes/square.js';
4

JS traingle.js x ...
modules1 > shapes > JS traingle.js > trang
1 export function trang(){
2   console.log("Area Of Traingle Function From Trang
   Function Circle.js");
3 }

JS square.js x ...
modules1 > shapes > JS square.js > sqr
1 export function sqr(){
2   console.log("Area Of Square Function From sq Function
   Circle.js");
3 }

JS circle.js x ...
modules1 > shapes > JS circle.js > circ
1 export function circ(){
2   console.log("Area Of Circle Function From Circ Function
   Circle.js");
3 }
```