Java Script

- Java script is a client side as well server side language.
- It is introduced in year 1995 by a person by name "Brendon EICH".
- Java Script is maintained by ECMA (European Computer Manufacture Association) from the year 1997.
- We have different version of JS like ES-1 to ES-6.
- With respect to front end JS is used for the following reason...
  1. To make the pages dynamic
  2. Validation
- Java Script code will be executed by JAVA SCRIPT ENGINE which is integrated with all browser's.

  Chrome
  I.E
  Firefox

NOTE:-

- JS is a scripting language
- Node JS is a JS library which is used to run the JS code in server.
- Mongo DB is used to store data
- Express JS is a framework is used to write business logic
- Angular, React are the framework of JS which are used to get front end

Contents:

Types of Java Script:

Based on the place where java Script is written we have **2 types of JS**

1. **Internal Java Script:**

   If the Java Script code is written in the same html page using Script tag. We call it has internal Java Script

   Ex: <script>----JS code--- </a>

2. **External Java Script:**

   If the Java Script code is written in a separate file with .js extension we called it as External Java Script.

   To link External JS following code will be written

   <script src="---path --"></script>

Java & Java Script are independent languages (No-relations)

Output Statements:

• document.write();	-	print in same line
• document.writeln();	-	print in same line & giving space
• console.log();	-	just for debug

NOTE:

- Semicolon is optional to end the Statements.
- It is not an error free language.
- Java Script is case Sensitive language & we can see error in console.

Concatenation can be done by using $\boxed{\textbf{(+) \& (,)}}$

• +	-	This operator the concat the content as it is.
• ,	-	This Operator will append space between 2 operands & concats

Keywords:

- All the keywords are written in lower case
- This are reserved words whose meaning will known to the Java Script engine
- Ex: let, if, else, continue, break etc…

Variables:

Variables are the container to hold some data.

Keyword:

- var – basic keyword from version 1
- let & const – keyword from version ES-6

Syntax: var/let/const varname; //syntax

```
Ex:     var a;          // declaration
        a=10;           //initialization
        a=20;           //re- initialization
        a=25.36         //re- initialization
        a=”hai”         //re- initialization
        var a;          //re-declaration is also possible
```

NOTE:

- Java Script is dynamically type checked language.
- If a variable is capable of storing different type of data then it is called as dynamically type checked language.

Java Script Features:

- Client side language
- Server side language it is used in server
- It is a scripting language
- It is case sensitive language
- Dynamically type checked language

let keyword:

```
let b;          //declaration
b=20;           //initialization
b=30.21         //re-initialization
b=’a’;          //re-initialization
b=true          //re-initialization
let b;          //re-initialization is not possible (we get error in console)
```

const keyword:

```
const c=10;     //declaration & initialization
```

- Both declaration & initialization has to be done in same line **const c=20;**
- In this keyword there is no re-initialization & re-declaration.

	Declaration	**initialization**	**Re-initialization**	**Re-declaration**
Var	Yes	yes	yes	Yes
Let	Yes	yes	yes	No
Const	Yes	yes	No	No

Operators:

1. **Arithmetic Operator: (+ , - , * , /, %)**
   - This are used to perform the arithmetic operations
   - In EXPRESSION evaluation +,- has to be given least priority compare to *,/,%
   - If same priority operators are present in an expression than we should follow left to right associativity.

```
let res = 10+20*2/4;
        document.write(res+"<br>");
```

   - **Ex:** 10+20/2*4
     10+10*4
     10+40
     50

NOTE:

- Division operator (/) will give complete result along with decimal values
  Ex: 10/2; //5
  10/3; //3.33333
- Modulus operator gives the remainder

2. Relational Operator:(< , > , <= , >= , == , != , === , !==)

- This operators are used to compare any 2 operands
- Relational operator always results in Boolean outputs(true/false)
- Equality Operator: (= =) It will check only for data
  10==10; //true
  10==”10”; //true

```
document.writeln("<b>Normality relational operators</b>"+"<br>");
        document.writeln(10==10);
        document.writeln(10!=11);
```

- Strict Equality Operator: (= = =) It will check for both data & type of the data
  10===10; //true
  10===”10”; //false

```
document.writeln("<b>Strictly relational operators</b>"+"<br>");
        document.writeln(10===10);
        document.writeln(10!==11);
```

3. **Logical Operators: (&& , ||)**
   - This operators are used to check more than 1 condition
   - Both input & output of logical operators is Boolean

Operand 1	**Operand 2**	**&&**	**||**
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

   - If all the conditions are true than “**logical &**” will be evaluated to true

- If any one of the conditions is true for logical or operations than it will be evaluated true
  - **Ex:** (10>20) && (10<20) || 5<20
    F && T || T
    F || T

4. **Bitwise Operators: (& , |)**

- It will convert Operands to Binary values & perform the operations, Result will converted back to decimal.

5. **Unary Operators: (++ , --)**

- **++(inc)**
  - Post Increment: (a++) (use value 1st , later inc)
  - Pre Increment: (++a) (1st inc , later use the value)
- **--(dec)**
  - Post Decrement:(a--) (use value 1st , later dec)
  - Pre Decrement:(--a) (1st dec , later use the value)

NOTE:

- Unary operators it has to be used only on variables declared using var or let, we should not used in const.

6. **Assignment Operators: (= , += , -= , *= , /= , %=)**

- a += 5; //a = a+5;
- a -= 5; //a = a-5;
- a *= 5; //a = a*5;

```
let a=10;
        document.write("a = "+a+"<br>");
    a += 15;
        document.write("a = "+a+"<br>");
```

7. **Turnery Operator:**

- (code) ? true : false;

```
let b = (10 < 20)
        b ? document.write("b = "+"stmt 1 : true") : document.write("b = "+"stmt 2 : false");
        document.write("<br>");
let c = (10 > 20)
        c ? document.write("c = "+"stmt 1 : true") : document.write("c = "+"stmt 2 : false");
        document.write("<br><br>");
```

8. **typeof Operator:**

- It is used to check what type of data variable is holding

```
 s = 10;
        document.write("S = "+s+" = "+typeof(s));
        document.write("<br>");
```

Control Statements:

It is used to control the flow of Execution

Control Statements

Branching Statements	Looping Statements
• If • if else • nested if • if else ladder • switch	• for • while • do while • for-in • for-of

Branching Statements:

```
//if else ladder
        let x = 0
        if(x > 0)
        {
            document.write(x+" = +ve Number");
        }
        else if(x < 0)
        {
            document.write(x+" = -ve Number");
        }
        else
        {
            document.write(x+" = neither +ve nor -ve");
        }
```

```
//nested if
        let a = 75;
        if(a > 0)
        {
            document.write(a+" = +ve");
            document.write("<br>");
            if(a % 2 == 0)
                document.write(a+" is even");
            else
                document.write(a+" is odd");
        }
        else if(a < 0)
        {
            document.write(a+" = -ve");
            document.write("<br>");
            if(a % 2 == 0)
                document.write(a+" is even");
            else
                document.write(a+" is odd");
        }
        else
        {
            document.write(a+" is neither +ve nor -ve");
        }
```

```
//switch
let ch="A";
        switch(ch)
        {
            case 1 : document.write("Int");
                break;
            case 'A' : document.write("Char");
                break;
            case "Hai" : document.write("String");
                break;
            case true : document.write("Boolean");
                break;
            default : document.write("default");
        }
```

Looping Statements:

For Loop:

```
for(initialization ; condition ; inc / dec )
{
        body of the loop
        ----
        ----
}
```

```
 //for loop
        for(let i=1 ; i<=5 ; i++)
        {
            document.write("hello"+"<br>");
        }
```

```
//sum of 1-10
        let sum = 0;
        for(let i=1 ; i<=10 ; i++)
        {
            document.write(i+" + ")
            sum += i;
        }
        document.write("<br>"+"Sum = "+sum);//55
```

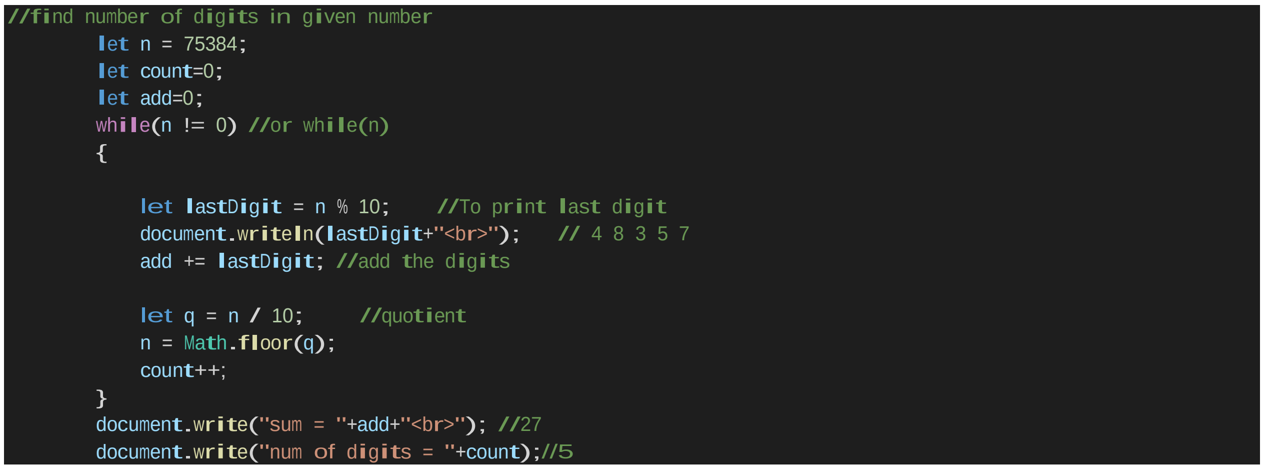
While Loop:

```
while(condition)
{
       ----
       ----
}
```

```
//while loop
        let m = 1;
        while(m <= 5)
        {
            document.writeln("Hello"+"<br>");
            m++;
        }
```

NOTE:

For Loop	While Loop
• If the number of iteration known use for loop	• If the number of iteration are not known use while loop



```
//find number of digits in given number
        let n = 75384;
        let count=0;
        let add=0;
        while(n != 0) //or while(n)
        {

            let lastDigit = n % 10;     //To print last digit
            document.writeln(lastDigit+"<br>");    // 4 8 3 5 7
            add += lastDigit; //add the digits

            let q = n / 10;     //quotient
            n = Math.floor(q);
            count++;
        }
        document.write("sum = "+add+"<br>"); //27
        document.write("num of digits = "+count);//5
```

Functions:

Functions are set of instructions to perform some task

Advantages:

- Code re-usability
- Modularity(Structure way of writing the program)

Syntax of declaring the functions:

```
        Function nameOfTheFunction(parameter1, parameter2, ….)
        {
                ----
                ----
        }
```

```
<head>
   <script>
        function fun1()     //declaration of the Function
        {
            //body of the function
            document.write("function1 is executing...!")
        }
        fun1();     //invoke/call the function
        fun1();
    </script>
</head>
<body>
    <h1>Functions Demo</h1>
</body>
```

or

```
<head>
    <script>
        function fun1()     //declaration of the Function
        {
            //body of the function
            document.write("function1 is executing...!")
        }
    </script>
</head>
<body>
    <h1>Functions Demo</h1>
    <script>
        fun1();     //invoke/call the function
        fun1();
    </script>
</body>
```

- NOTE: Functions will not be executed unless it will invoke or call

Functions with Parameters:

```
<script>
        function fun2(a)
        {
            document.write("a = "+a+"<br>")
            document.write("function-2 is executed <br>")
        }
        fun2(10);       //a = 10
        fun2(10.12);    //a = 10.12
        fun2(true);     //a = true
        fun2("hai");    //a = hai
        fun2();         //a = undefined
</script>
```

NOTE:

- For a function with parameter we can call the function without passing arguments
- If we are not passing the arguments it will take the default value which is undefined
- We can change default value by assigning some value to the parameters
  function fun(a=100)

```
<script>
        function fun3(a=100)
        {
            document.write("a = "+a+"<br>")
            document.write("function-3 is executed <br>")
        }
        fun3();   //a=100
</script>
```

```
<script>
         function fun4(a=1,b=2,c=3)
        {
            document.write("a = "+a+"<br>")
            document.write("b = "+b+"<br>")
            document.write("c = "+c+"<br>")
            document.write("function-4 is executed <br>")
        }
         fun4(10,20,30)            //a=1 b=2 c=3
</script>
```

```
<script>
//factorial program
         function factorial(num =0)
        {
            let fact = 1;
            while(num)
            {
                fact = fact * num;
                num--;
            }
            document.writeln(fact);
        }
         factorial(5);    //120
</script>
```

Functions some return values:

```
<script>
          function fac5()
        {
            document.write("function-5 is executed<br>");
            return 100;
        }
        let x = fun5();
        document.writeln(x)
</script>
```

Functions some parameters & some return values:

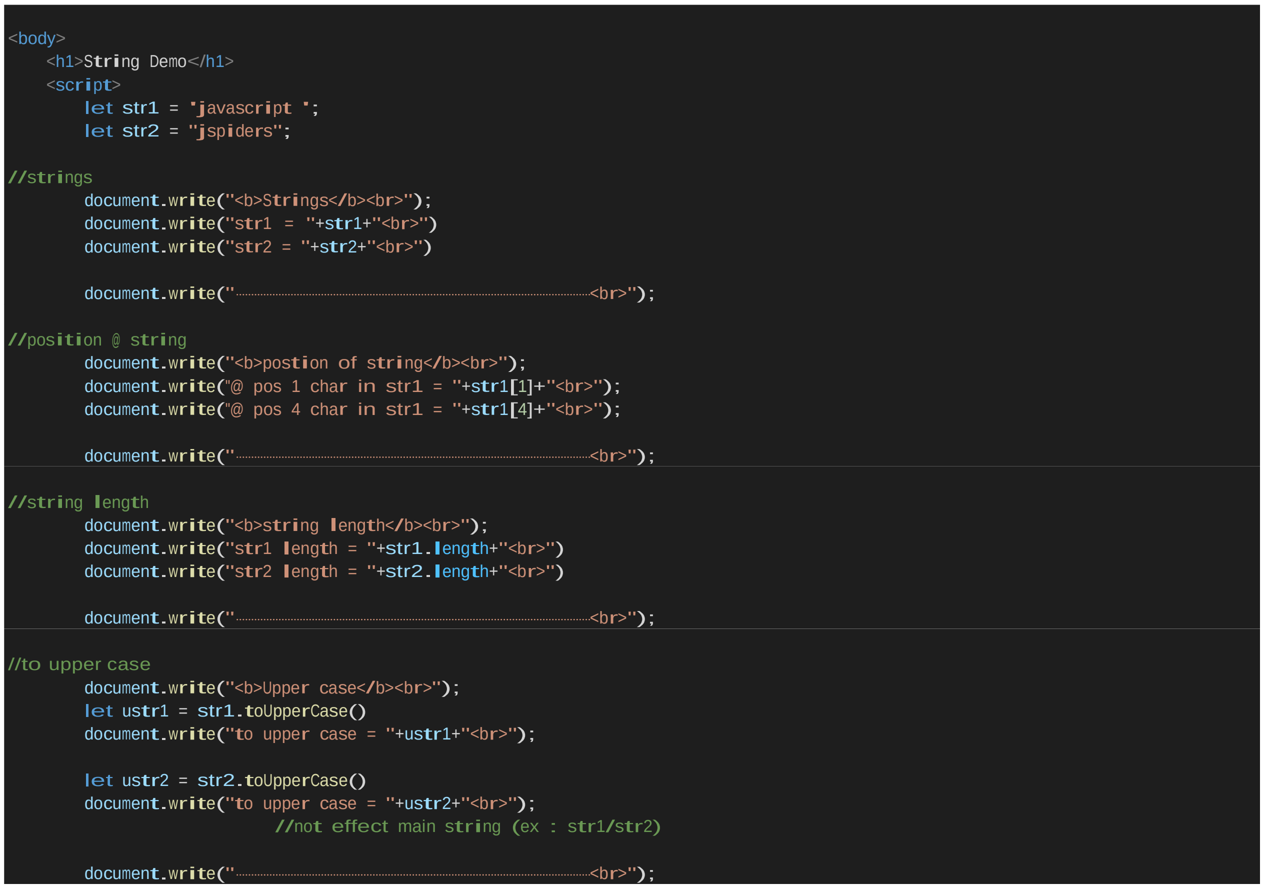
```
<script>
         function fac6(a)
        {
            document.write("function-6 is executed<br>");
            return 10;
        }
         let y = fun6();

        document.writeln(y);
</script>
```

NOTE: A function can return only one value where as accepts many parameters

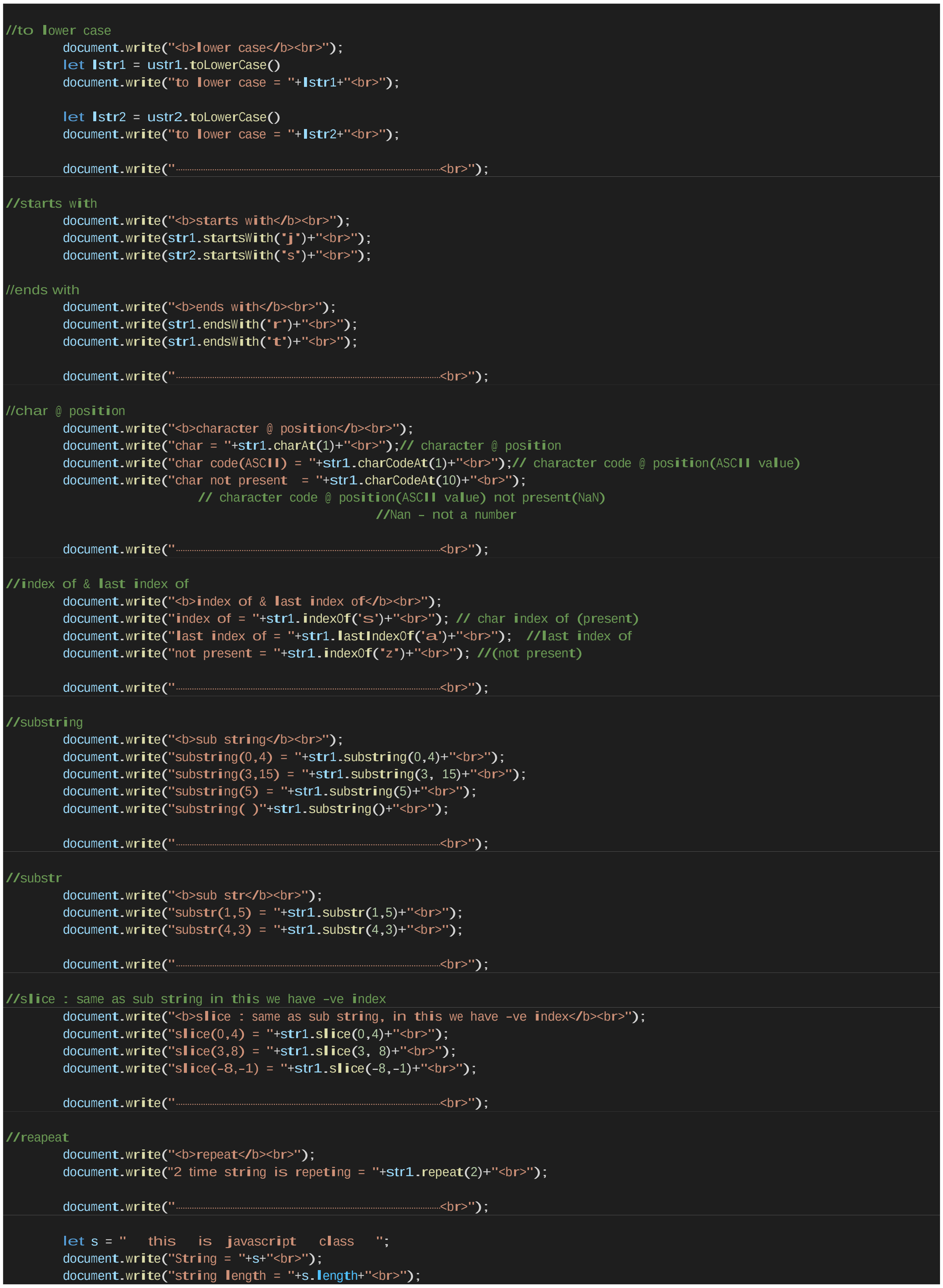
STRINGS:

String and Methods:

- To store a group of characters we will use strings
- To declare the strings we will follow these ways ' '/" "/` `
    - i. toLowerCase()
    - ii. toUpperCase()
    - iii. SubString(para1,para2)
    - iv. substr(para1,para2)
    - v. startswith(para1)//boolean value
    - vi. endsWith(para1)//boolean value
    - vii. charAt(para1)
    - viii. charCodeAt(para1)
    - ix. split(para1)
    - x. slice(para1,para2)
    - xi. indexOf()
    - xii. lastIndexOf()
    - xiii. trim()
- NAN=NOt a Number
- sub String- in main string to get a part of string start index,ending index
- starting index,ending index-1



```
<body>
    <h1>String Demo</h1>
    <script>
        let str1 = 'javascript ';
        let str2 = "jspiders";

//strings
        document.write("<b>Strings</b><br>");
        document.write("str1  =  "+str1+"<br>")
        document.write("str2 = "+str2+"<br>")

        document.write("..........................................................................<br>");

//position @ string
        document.write("<b>postion of string</b><br>");
        document.write("@ pos 1 char in str1 = "+str1[1]+"<br>");
        document.write("@ pos 4 char in str1 = "+str1[4]+"<br>");

        document.write("..........................................................................<br>");

//string length
        document.write("<b>string length</b><br>");
        document.write("str1 length = "+str1.length+"<br>")
        document.write("str2 length = "+str2.length+"<br>")

        document.write("..........................................................................<br>");

//to upper case
        document.write("<b>Upper case</b><br>");
        let ustr1 = str1.toUpperCase()
        document.write("to upper case = "+ustr1+"<br>");

        let ustr2 = str2.toUpperCase()
        document.write("to upper case = "+ustr2+"<br>");
                            //not effect main string (ex : str1/str2)

        document.write("..........................................................................<br>");
```



```
//to lower case
        document.write("<b>lower case</b><br>");
        let lstr1 = ustr1.toLowerCase()
        document.write("to lower case = "+lstr1+"<br>");

        let lstr2 = ustr2.toLowerCase()
        document.write("to lower case = "+lstr2+"<br>");

        document.write("-------------------------------------------------------------------------------------------------<br>");

//starts with
        document.write("<b>starts with</b><br>");
        document.write(str1.startsWith("j")+"<br>");
        document.write(str2.startsWith("s")+"<br>");

//ends with
        document.write("<b>ends with</b><br>");
        document.write(str1.endsWith('r')+"<br>");
        document.write(str1.endsWith('t')+"<br>");

        document.write("-------------------------------------------------------------------------------------------------<br>");

//char @ position
        document.write("<b>character @ position</b><br>");
        document.write("char = "+str1.charAt(1)+"<br>");// character @ position
        document.write("char code(ASCII) = "+str1.charCodeAt(1)+"<br>");// character code @ position(ASCII value)
        document.write("char not present  = "+str1.charCodeAt(10)+"<br>");
                                // character code @ position(ASCII value) not present(NaN)
                                                                //Nan - not a number

        document.write("-------------------------------------------------------------------------------------------------<br>");

//index of & last index of
        document.write("<b>index of & last index of</b><br>");
        document.write("index of = "+str1.indexOf('s')+"<br>"); // char index of (present)
        document.write("last index of = "+str1.lastIndexOf('a')+"<br>");  //last index of
        document.write("not present = "+str1.indexOf("z")+"<br>"); //(not present)

        document.write("-------------------------------------------------------------------------------------------------<br>");

//substring
        document.write("<b>sub string</b><br>");
        document.write("substring(0,4) = "+str1.substring(0,4)+"<br>");
        document.write("substring(3,15) = "+str1.substring(3, 15)+"<br>");
        document.write("substring(5) = "+str1.substring(5)+"<br>");
        document.write("substring( )"+str1.substring()+"<br>");

        document.write("-------------------------------------------------------------------------------------------------<br>");

//substr
        document.write("<b>sub str</b><br>");
        document.write("substr(1,5) = "+str1.substr(1,5)+"<br>");
        document.write("substr(4,3) = "+str1.substr(4,3)+"<br>");

        document.write("-------------------------------------------------------------------------------------------------<br>");

//slice : same as sub string in this we have -ve index
        document.write("<b>slice : same as sub string, in this we have -ve index</b><br>");
        document.write("slice(0,4) = "+str1.slice(0,4)+"<br>");
        document.write("slice(3,8) = "+str1.slice(3, 8)+"<br>");
        document.write("slice(-8,-1) = "+str1.slice(-8,-1)+"<br>");

        document.write("-------------------------------------------------------------------------------------------------<br>");

//reapeat
        document.write("<b>repeat</b><br>");
        document.write("2 time string is repeting = "+str1.repeat(2)+"<br>");

        document.write("-------------------------------------------------------------------------------------------------<br>");

        let s = "   this   is  javascript   class   ";
        document.write("String = "+s+"<br>");
        document.write("string length = "+s.length+"<br>");
```

```
//trim : trim starting space & ending space
        let s1 = s.trim();
        document.write("trim = "+s1+"<br>");
//split
        document.write("split = "+s1.split(" ")+"<br>");

        document.write("------------------------------------------------------------<br>");

    </script>
</body>
```

Immutability:

- String is immutable
- On the string if we perform some changes using inbuilt methods, all the changes will be effected on new string, Original string will be unchanged this behavior is called as immutability

```
//immutability
        let x1 = "abc";
        let x2 = x1.toUpperCase()
        if(x1 == x2)
            document.write("Immutable");
        else
            document.write("not Immutable"); // not Immutable
```

- If we convert string which has other than digits to number we will get NaN(not a number)

Example Programs:

1. Print A to Z (lower case & upper case)

```
//print A to Z
             document.write("<b>Print A to Z </b><br>");
        for(let i=65 ; i<=90 ; i++)
        {
            document.write(String.fromCharCode(i));//ABCDEFGHIJKLMNOPQRSTUVWXYZ
        }
//print a to z
        for(let i=97 ; i<=122 ; i++)
        {
            document.write(String.fromCharCode(i));//abcdefghijklmnopqrstuvwxyz
        }
```

ARRAY'S & It's Methods:

- Arrays are to store the data into single entity.
- Arrays are heterogeneous & grow able in nature.

```
//Example
        let arr1 = [10,10.23,true,"Hello"]
            document.write(arr1+"<br>");     //10,10.23,true,Hello
            document.write(arr1[0]+"<br>"); //10
            document.write(arr1[1]+"<br>"); //10.23
            document.write(arr1[2]+"<br>"); //true
            document.write(arr1[3]+"<br>"); //Hello
            document.write(arr1[4]+"<br>"); //undefined
```

```
//we can change the values
        arr1[0]=100;
        document.write(arr1[0]+"<br>")  //100
        arr1[4]=500;
        document.write(arr1[4]+"<br>")  //500
        document.write(arr1+"<br>");    //100,10.23,true,Hello,500

        arr1[6]=50;
        document.write(arr1[6]+"<br>"); //50
        document.write(arr1+"<br>");    //100,10.23,true,Hello,500,undefined,50

        document.write("array length = "+arr1.length);  //7
```

```
//Display all the arrays
        for(let i=0 ; i<arr1.length ; i++)
        {
            document.writeln("arr1["+i+"] = "+arr1[i]+"<br>");
        }
            // arr1[0] = 100
            // arr1[1] = 10.23
            // arr1[2] = true
            // arr1[3] = Hello
            // arr1[4] = 500
            // arr1[5] = undefined
            // arr1[6] = 50
```

```
//Display only Integer
            for(let i=0 ; i<arr1.length ; i++)
            {
                if(typeof(arr1[i])=="number")
                {
                    document.writeln("arr1["+i+"] = "+arr1[i]+"<br>");
                }
            }
                // arr1[0] = 100
                // arr1[1] = 10.23
                // arr1[4] = 500
                // arr1[6] = 50
```

```
//Display only String
            for(let i=0 ; i<arr1.length ; i++)
            {
                if(typeof(arr1[i])=="string")
                {
                    document.writeln("arr1["+i+"] = "+arr1[i]+"<br>");
                }
            }
                //Hello
```

<u>Methods of Arrays:</u>

- push(para1,para2,…)
- pop()
- unshift(para1,para2,…)
- shift()
- splice(para1,para2,para3…..ParaN)
  - para1 : index
  - para2 : no of elements to be removed
  - para3..paraN : elements to be added
- indexOf|(para1)
- slice(arg1, arg2)
- join(para1)

<u>Push:</u> Push Method will add the elements at the last & written new length

```
//Push Method example
            arr1.push(1000,2000);
            document.write("After push method  = "+arr1+"<br>");    //100,10.23,true,Hello,500, ,50,1000,2000
            document.write("array length = "+arr1.length+"<br>");    //9
```

<u>Pop:</u> Pop method will remove the element present in last

```
//pop Method
            arr1.pop();
            document.write("After pop method  = "+arr1+"<br>");      //100,10.23,true,Hello,500,,50,1000
            document.write("array length = "+arr1.length+"<br>");    //8
```

<u>Unshift:</u> add the elements at the 1st & written new length

```
        //un-shift
            arr1.unshift("hai","hello")
            document.write("After <b><u>unshift</u></b> method  = "+arr1+"<br>");
                     //hai,hello,100,10.23,true,Hello,500,,50,1000
            document.write("array length = "+arr1.length+"<br>");    //10
```

<u>Shift:</u> Remove the elements at the 1st.

```
//shift
        arr1.shift()
        document.write("After <b><u>shift</u></b> method  = "+arr1+"<br>");  //hello,100,10.23,true,Hello,500,,50,1000
        document.write("array length = "+arr1.length+"<br>");    //9
```

<u>Splice:</u> Adding & removing the elements in between

```
//splice
            let removeEle = arr1.splice(1,2,"new1","new2","new3")
            document.write("After <b><u>splice</u></b> method  = "+arr1+"<br>");
                //hello,new1,new2,new3,true,Hello,500,,50,1000
            document.write("Removed elements are = "+removeEle+"<br>")  //100,10.23
            document.write("array length = "+arr1.length+"<br>");    //10
```

```
            let removeEle1 = arr1.splice(4,0,"new4","new5","new6")
            document.write("After <b><u>splice</u></b> method  = "+arr1+"<br>");
                // hello,new1,new2,new3,new4,new5,new6,true,Hello,500,,50,1000
            document.write("Removed elements are = "+removeEle1+"<br>"); //not removed
            document.write("array length = "+arr1.length+"<br>");//13
            let removeEle2 = arr1.splice(3,4);
            document.write("After <b><u>splice</u></b> method  = "+arr1+"<br>");
            // hello,new1,new2,true,Hello,500,,50,1000
            document.write("Removed elements are = "+removeEle2+"<br>"); //new3,new4,new5,new6
            document.write("array length = "+arr1.length+"<br>");//9
```

IndexOf:

- If value present it will return index value or else it will return -1 value

```
 //indexof
            document.write(arr1.indexOf("new2"+"<br>")); //2
            document.write(arr1.indexOf(2000+"<br>")); //-1
```

Slice:

```
//Slice
          document.write("<b><u>slice</u></b> method  = "+arr1.splice(2,5)+"<br>");     //new2,true,Hello
          document.write("After <b><u>splice</u></b> method  = "+arr1+"<br>");
               //hello,new1,new2,true,Hello,500,,50,1000
```

Joins:

```
//joins
            document.write("<b><u>joins</u></b> method  = "+arr1.join(" & "));
                //hello & new1 & new2 & true & Hello & 500 & & 50 & 1000
```

Ex Program:

```
//Example Program
    document.writeln("<h2> Ex Program:</h2>")
        let arr5 = [10,20,30,40,50,80];
        document.write("Array Elemets = "+arr5+"<br>");
        let newElem = 500;
        let index = arr5.indexOf(newElem);
        if(index === -1)
        {
            document.write("Element "+newElem+" is not Present : <br>");
            arr5.splice("Adding = "+3,0,newElem)
        }
        else
        {
            document.write("Element "+newElem+" is present : ");
            arr5.splice(index,1)
        }
        document.write("After Adding : "+arr5)
        /*  Array Elemets = 30,20,10,70,40,50,80,60
            Element 500 is not Present :
            After Adding : 500,30,20,10,70,40,50,80,60
        */
```

Sort:

```
 //sort
        function myOwnSort(a,b)
        {
            return a-b;
        }
        document.write("After sorting in assending = "+arr5.sort(a,b)+"<br>");//10,20,30,40,50,60,70,80,500
```

Objects:

- Objects are real world entities which has its own states and behavior
- Here states represent the properties of objects which can be represented using Data members.
- Behavior represents functionality of an object using methods.
  Ex: car (States: - name, color, max & min Speed etc…)
  (Behavior: - Start engine, apply break, apply gear etc…)
- In java script we can create the objects using following 3 types
  - Direct literals
  - New Keyword
  - Constructor functions
- In the Objects the data will be stored in the form of name & value pairs.

Direct Literals:

```
let/var/const = {
                    property 1 : value 1,
                    property 2 : value 2,
                    -----
                    property n : value n,
                 };
```

```
//Direct Literals
        let car1 =
        {
            name : "KIA",
            model : 2020,
            color : "black red",
            milage : 15
        };
        console.log(typeof(car1)); //object
        console.log(car1); //name: "KIA", model: 2020, color: "black red', milage: 15
```

- To access the data from the object we use following 2 ways
  - Dot operator (.)
  - Sub Script operator([ ])

```
//dot operator
        console.log(car1.name);      //KIA
        console.log(car1.color);     //black red
        console.log(car1.model);     //2020
        console.log(car1.milage);    //15
```

```
//sub script operator
        console.log(car1["name"]);  //KIA
        console.log(car1["color"]); //black red
        console.log(car1["model"]); //2020
        console.log(car1["milage"]);//15
```

```
//change some property
        car1.milage = 13.5;
        console.log(car1);   //name: "KIA", model: 2020, color: "black red", milage: 13.5
```

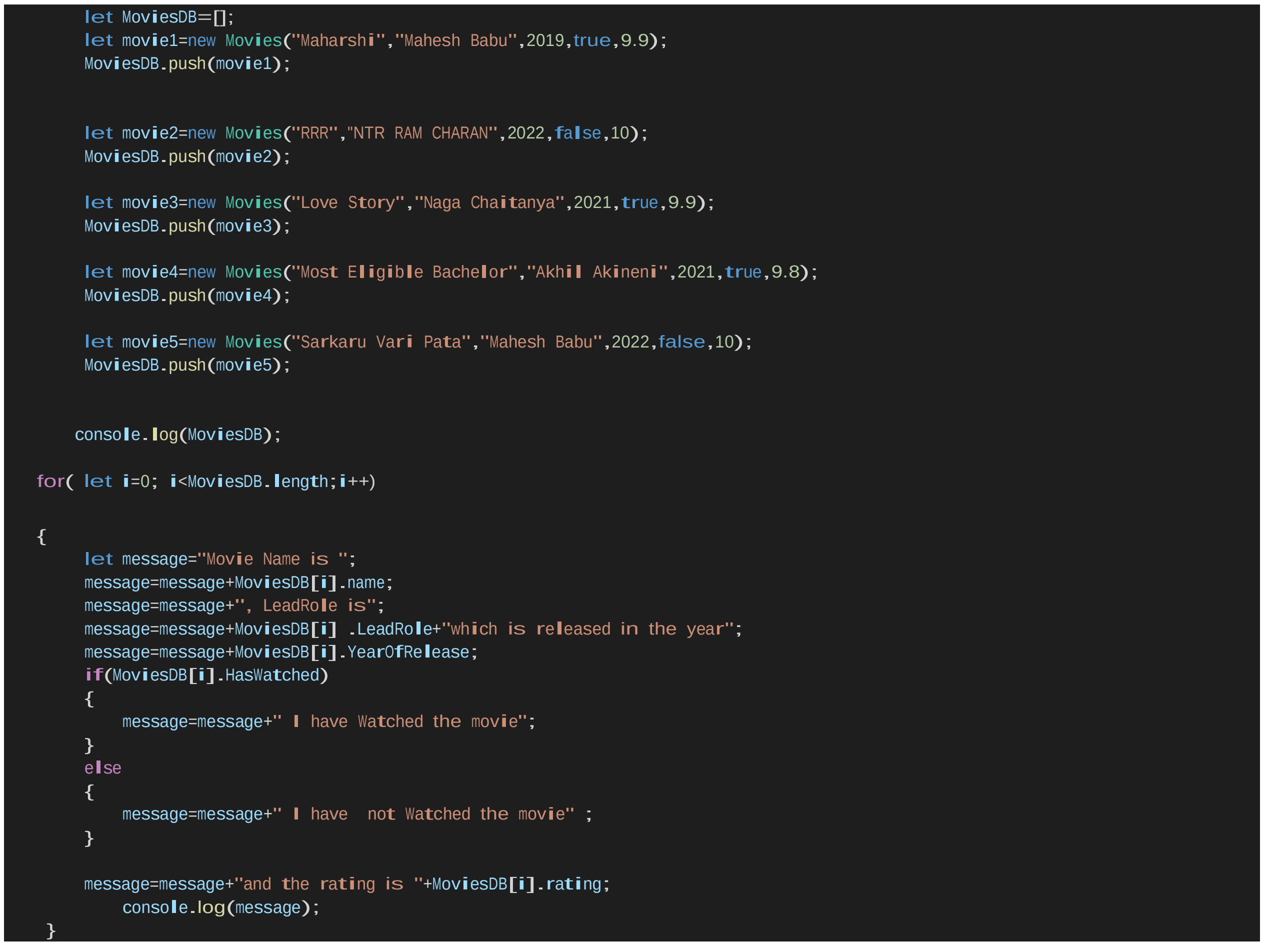
```
//add some property
        car1["regNo"] = "KA 00 AB 0000';
        console.log(car1);    //name: "KIA", model: 2020, color: "black red", milage: 13.5, regNo: 'KA 00 AB 0000'
```

```
//delete/remove the property
        delete car1.color;
        console.log(car1);  //name: "KIA", model: 2020, milage: 13.5, regNo: "KA 00 AB 0000"
```

New Keyword:

Constructor functions:

```
//constructor functions
       let student1 =
        {
           firstname :"dinesh",
           lastname:"reddy",
           marks:76

       }

       console.log(typeof(student1));
       console.log(student1);

       //object creation using the new keyword

     let car2 = new Object();

     console.log(typeof(car2));

     car2.name = "skoda";
     car2.model = 2021;
     car2.color = "blue";

     let person = new Object();
     console.log(typeof(person));
     console.log("<br>");
     person.name = "hari";
     person.age = 20;
     person.weight = 30;
     console.log(person["name"]);
     console.log("<br>");
     console.log(person["age"]);
     console.log("<br>");
     console.log(person["weight"]);
     console.log("<br>");

       function Car()
       {
           this.name="KIA";
           this.color="White";
           this.model=2019;
       }

       let car3=new Car();
       console.log(typeof(car3));
       console.log(car3);

       let car4=new Car();
       console.log(car4);

       function Car(nm,color,model)
       {
           this.name=nm;
           this.color=color;
           this.model=model;
       }

       let car5=new Car("BMW","black",1997);
       console.log(car5)

       function Movies(name,LeadRole,YearOfRelease,HasWatched,rating)
       {
           this.name =name;
           this.LeadRole=LeadRole;
           this.YearOfRelease=YearOfRelease;
           this.HasWatched=HasWatched;
           this.rating=rating;
       }
```

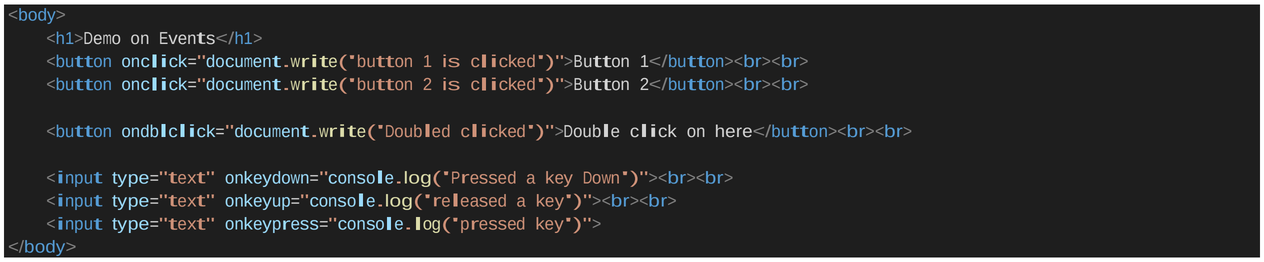


```
     let MoviesDB=[];
     let movie1=new Movies("Maharshi","Mahesh Babu",2019,true,9.9);
     MoviesDB.push(movie1);


     let movie2=new Movies("RRR","NTR RAM CHARAN",2022,false,10);
     MoviesDB.push(movie2);

     let movie3=new Movies("Love Story","Naga Chaitanya",2021,true,9.9);
     MoviesDB.push(movie3);

     let movie4=new Movies("Most Eligible Bachelor","Akhil Akineni",2021,true,9.8);
     MoviesDB.push(movie4);

     let movie5=new Movies("Sarkaru Vari Pata","Mahesh Babu",2022,false,10);
     MoviesDB.push(movie5);


    console.log(MoviesDB);

for( let i=0; i<MoviesDB.length;i++)


{
     let message="Movie Name is ";
     message=message+MoviesDB[i].name;
     message=message+", LeadRole is";
     message=message+MoviesDB[i] .LeadRole+"which is released in the year";
     message=message+MoviesDB[i].YearOfRelease;
     if(MoviesDB[i].HasWatched)
     {
         message=message+" I have Watched the movie";
     }
     else
     {
         message=message+" I have  not Watched the movie" ;
     }

     message=message+"and the rating is "+MoviesDB[i].rating;
         console.log(message);
 }
```

Events:

- Events are the Operations performed on web page like clicking, selecting, coping etc…
- Always Events has to be used as a attributes on HTML elements.

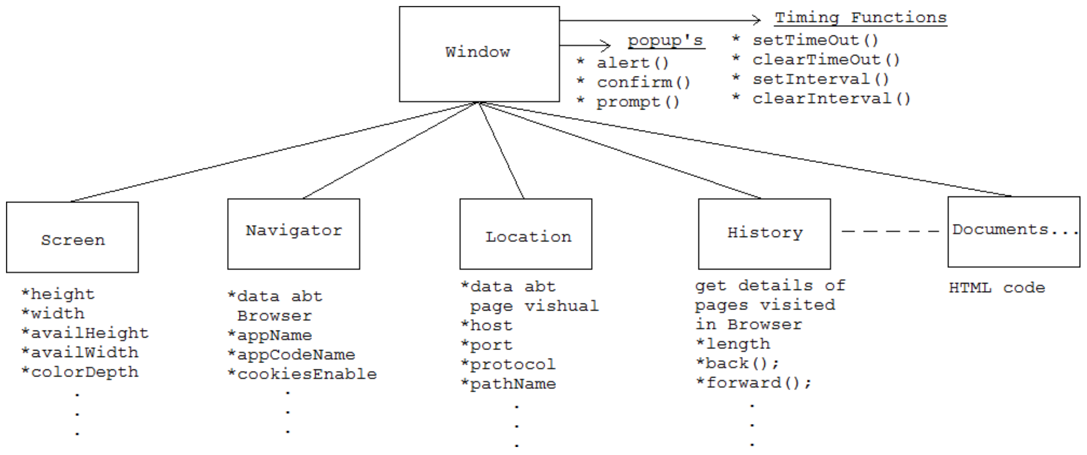
  Ex:
    - onclick = " "
    - onkeyup = " "
    - onkeydown = " "
    - onkeypress = " "
    - ondblclick = " "
    - oncopy = " "
    - onpaste = " "



```
<body>
    <h1>Demo on Events</h1>
    <button onclick="document.write('button 1 is clicked')">Button 1</button><br><br>
    <button onclick="document.write('button 2 is clicked')">Button 2</button><br><br>

    <button ondblclick="document.write('Doubled clicked')">Double click on here</button><br><br>

    <input type="text" onkeydown="console.log('Pressed a key Down')"><br><br>
    <input type="text" onkeyup="console.log('released a key')"><br><br>
    <input type="text" onkeypress="console.log('pressed key')">
</body>
```

Browser Object Model (BOM):

- Browser is represented in the form of window java script object.
- In depth study of Browser is called as Browser Object Model(BOM)
- To work with Browser using Java Script we will use window Object.
- Window object has many Methods, Properties & Objects inside it.

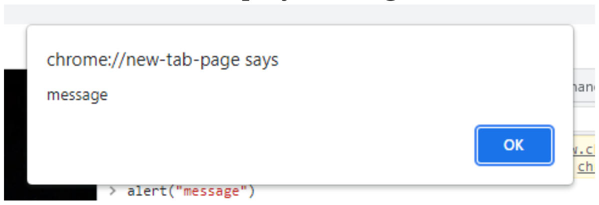


NOTE:

- Window is the default object in java script
- All the variables & Methods defined by User will be under the control of window object.
- Using window object name to access the properties of window object is optional.
- window.navigator (or) navigator in (console)

<u>popup's:</u>

- <u>alert();</u>
  - It is used to display message to end user.

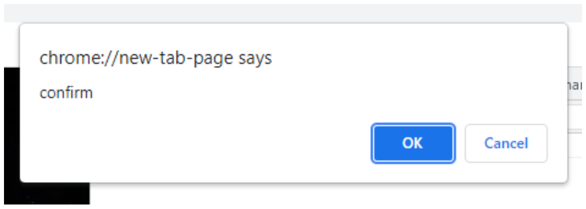


chrome://new-tab-page says

message

```
> alert("message")
```

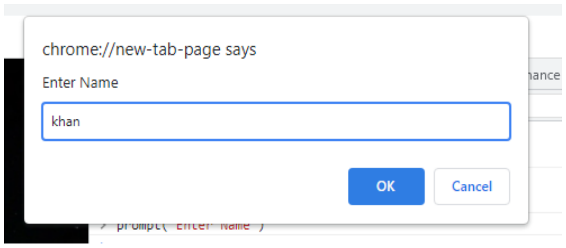
- <u>confirm();</u>
  - It is used to take additional confirmation from the user
  - Confirm method will return boolean values
  - If OK button is pressed it will return true if CANCLE button is pressed it will return false values.



chrome://new-tab-page says

confirm

- <u>prompt();</u>
  - It is used to take the input from the user.
  - It will return the value entered in the input field in the form of string if OK is pressed else it will return NULL

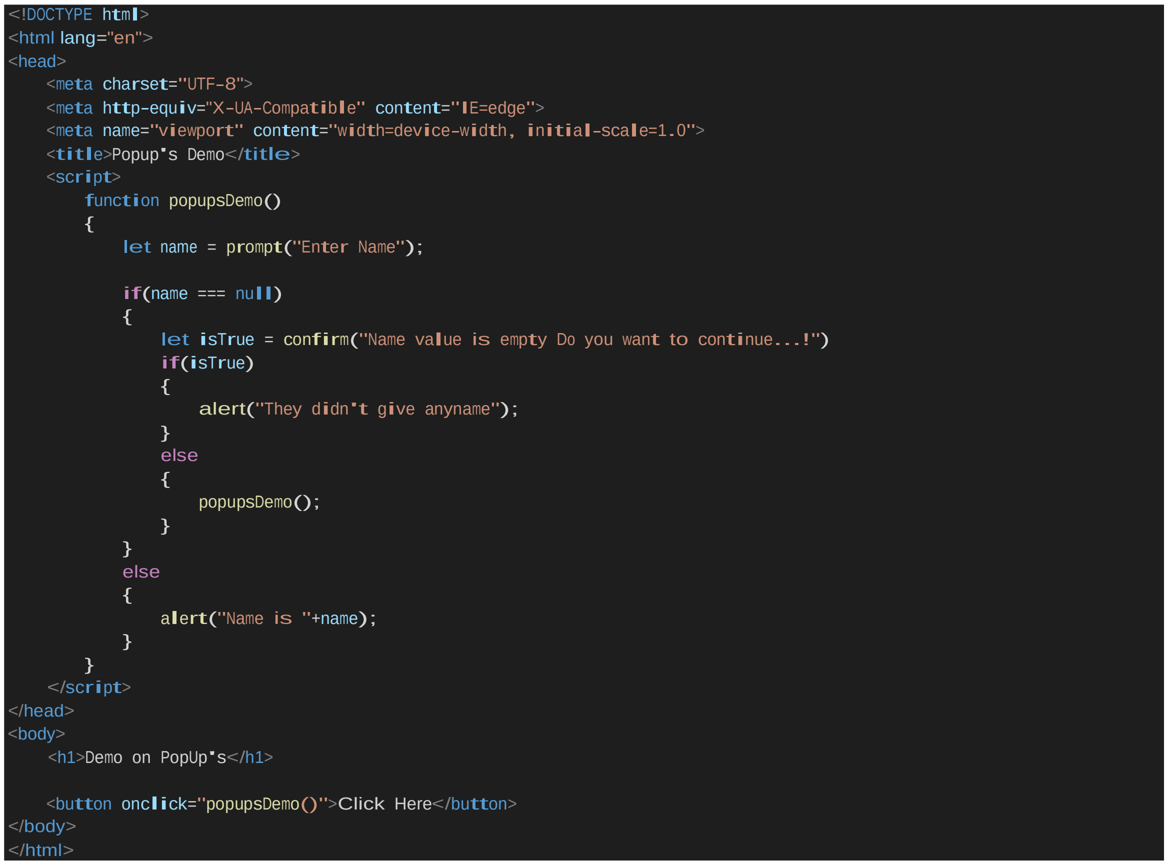


chrome://new-tab-page says

Enter Name

khan

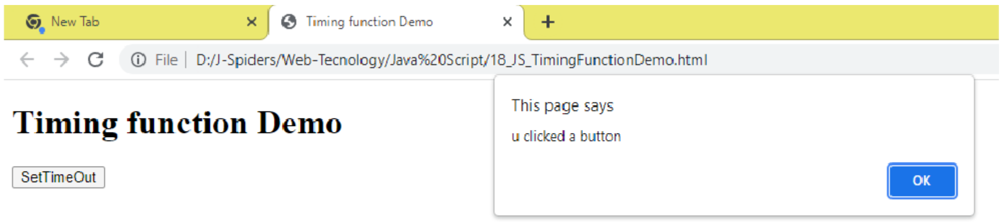
```
> prompt("Enter Name")
```



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Popup's Demo</title>
    <script>
        function popupsDemo()
        {
            let name = prompt("Enter Name");

            if(name === null)
            {
                let isTrue = confirm("Name value is empty Do you want to continue...!")
                if(isTrue)
                {
                    alert("They didn't give anyname");
                }
                else
                {
                    popupsDemo();
                }
            }
            else
            {
                alert("Name is "+name);
            }
        }
    </script>
</head>
<body>
    <h1>Demo on PopUp's</h1>

    <button onclick="popupsDemo()">Click Here</button>
</body>
</html>
```

Timing Function Demo:

```
<body>
    <h1>Timing function Demo</h1>

    <button onclick="alert('u clicked a button")">SetTimeOut</button>
</body>
```


New Tab
Timing function Demo
File | D:/J-Spiders/Web-Tecnology/Java%20Script/18_JS_TimingFunctionDemo.html
SetTimeOut
OK

Timing function Demo

This page says

u clicked a button

- Set Time Out
  - This method is used to give delay
  - Set time out function will return some unique value which helps to stop the execution of set time out method

```
<script>
        function fun1()
        {
            console.log("Fun1 is Executed...!")
        }
</script>

<!-- SetTimeOut -->
    <button onclick="setTimeout(fun1,2000)">SetTimeOut</button> <!-- 2000=2ms -->
```

- Clear Time Out
  - It is used to stop the execution of set Time out
  - It will take one argument & the argument is returned value of set Timed out

```
<!-- SetTimeOut -->
    <button onclick="a = setTimeout(fun1,2000)">SetTimeOut-2</button> <!-- 2000=2ms -->

<!-- ClearTimeOut -->
    <button onclick="clearTimeout(a);">stop setTimeout</button> <!-- a = variable -->
```

- Set Interval
  - It is used to execute the function @ regular interval of time

```
<script>
        function fun2()
        {
            console.log("Fun2 is Executed...!");
            console.log("hai");
        }
</script>
<!-- setInterval -->
    <button onclick="setInterval(fun2,2000)">Start Interval</button>
```

- Clear Interval

```
<!-- setInterval -->
    <button onclick="b = setInterval(fun2,2000)">Start Interval</button>

<!-- clearInterval -->
    <button onclick="clearInterval(b)">Stop setInterval</button>
```

Programs:

1. Print Numbers

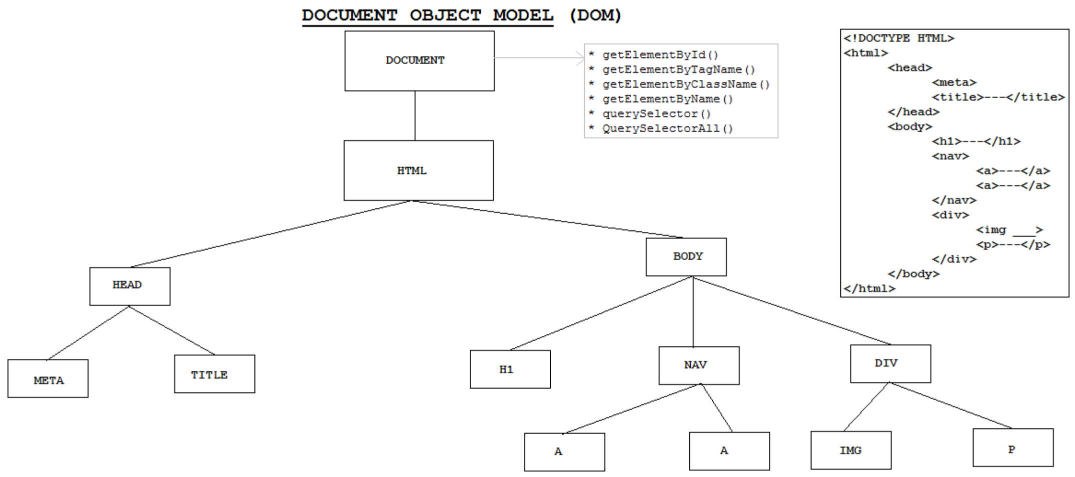
```
 <!-- Print Number -->
    <script>
        var num = 1;
        function printNum()
        {
            console.log(num);
            num++;
        }
    </script>
<!-- print Number  -->
    <button onclick="c = setInterval(printNum,1000)">Print Num</button>
    <button onclick="clearInterval(c)">Stop Print Num</button>
```

2. Take Number From User where to start and end. Print the Number

```
<!-- Print Number -->
    <script>
         var StartNum = prompt("Enter Start Number");
        var EndNum = prompt("Enter End Number");
        function printNum()
        {
            if(StartNum <= EndNum)
            {
                console.log(StartNum);
                StartNum++;
            }
            else
            {
                clearInterval(c);
            }
        }
    </script>

<!-- print Number  -->
    <button onclick="c = setInterval(printNum,1000)">Print Num</button><!-- 1000=1ms -->
```

Document Object Model: (DOM)

- Under the window object we have document object which helps to control HTML document
- Under document Object according to HTML code a structure will be created which is called as DOM
- Whenever HTML page loads DOM is created by Browser.



```
<!DOCTYPE HTML>
<html>
        <head>
                <meta>
                <title>---</title>
        </head>
        <body>
                <h1>---</h1>
                <nav>
                        <a>---</a>
                        <a>---</a>
                </nav>
                <div>
                        <img ___>
                        <p>---</p>
                </div>
        </body>
</html>
```

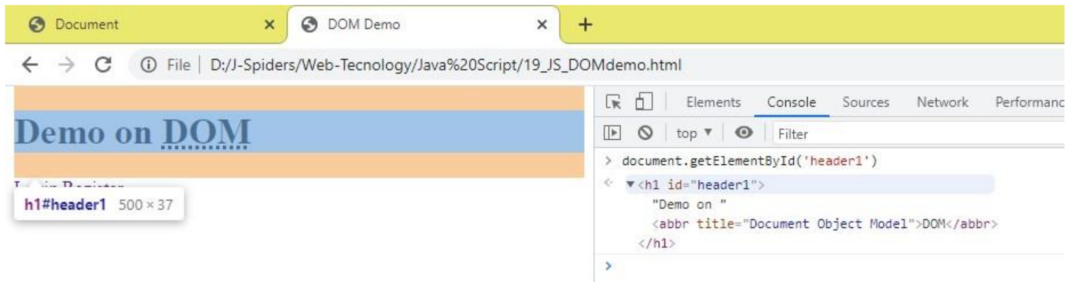
- In DOM html elements will be treated as Java Script Objects, Attributes of HTML elements will become properties of that object
- DOM is used for Dynamically changing the HTML pages by doing some manipulation

DOM Manipulation:

- To make the pages as Dynamic we will change the DOM which is turned as DOM Manipulation
- To do DOM Manipulation fallowing steps as to be used:
  - Select the HTML element which has to be changed
    - To select the HTML elements fallowing methods will be used which are present in Document object.
      - ✓ getElementById( )
      - ✓ getElementByTagName( )
      - ✓ getElementByClassName( )
      - ✓ getElementByName( )
      - ✓ querySelector( )
      - ✓ querySelectorAll( )
  - Do the Changes
    - ✓ Changes the content
    - ✓ Change the CSS Style
    - ✓ Add & remove the class
    - ✓ Change the Attributes
    - ✓ Add & remove HTML elements.

❖ getElementById Method:

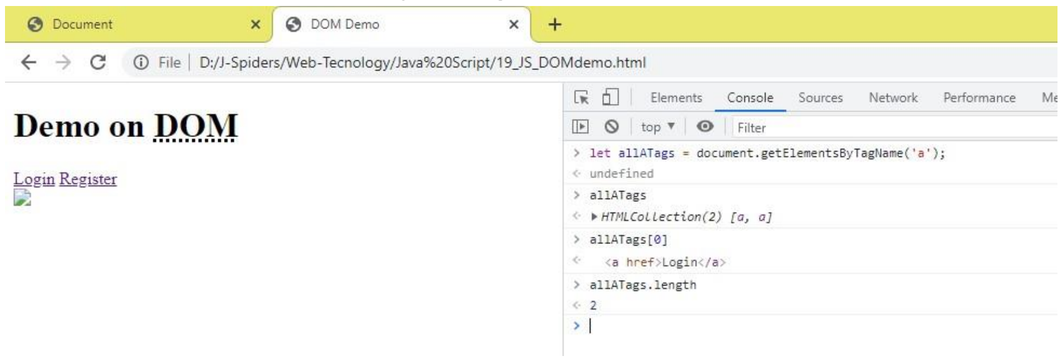
- This method is used to select HTML elements based on the ID name.
- ID name has to be passed as an argument for this method.
- This method will return an element with whatever the ID name we have been passed.
- This method will write only one element since ID's are Unique.



```
> document.getElementById('header1')
< ▼<h1 id="header1">
      "Demo on "
      <abbr title="Document Object Model">DOM</abbr>
   </h1>
>
```

❖ getElementsByTagName( )

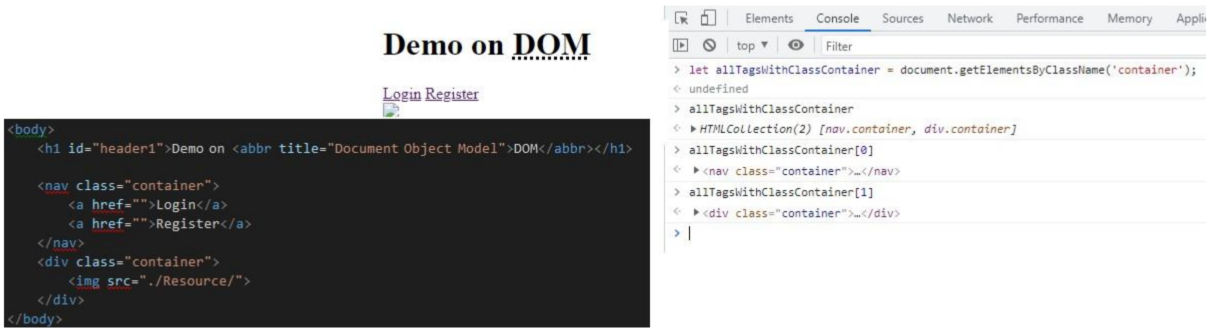
- This method is used to select HTML
- To select the elements using tag name pass tag name as a argument for this method in the form of string
- This method will return an array of Tags



```
> let allATags = document.getElementsByTagName('a');
< undefined
> allATags
< ▶HTMLCollection(2) [a, a]
> allATags[0]
<   <a href>Login</a>
> allATags.length
< 2
> |
```

❖ getElementByClassName( )

- This Method Helps to select HTML elements Based on the Class Name
- To Select the Elements Pass Class Name as an Argument in the form of Strings.
- This Method will return all the Matched elements in the form of Array

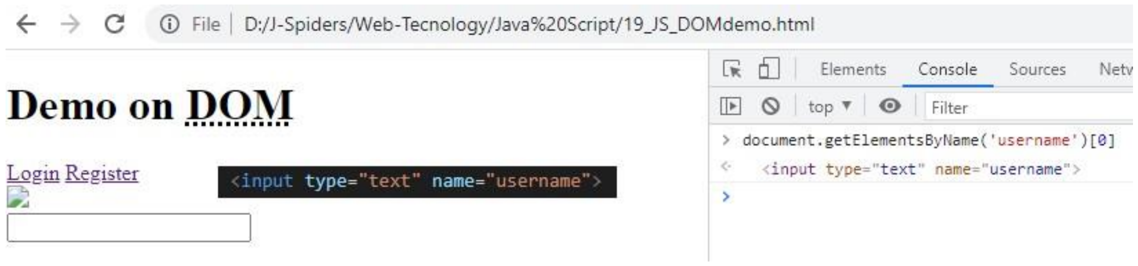


```
<body>
    <h1 id="header1">Demo on <abbr title="Document Object Model">DOM</abbr></h1>

    <nav class="container">
        <a href="">Login</a>
        <a href="">Register</a>
    </nav>
    <div class="container">
        <img src="./Resource/">
    </div>
</body>
```

```
> let allTagsWithClassContainer = document.getElementsByClassName('container');
< undefined
> allTagsWithClassContainer
< ▶HTMLCollection(2) [nav.container, div.container]
> allTagsWithClassContainer[0]
< ▶<nav class="container">…</nav>
> allTagsWithClassContainer[1]
< ▶<div class="container">…</div>
> |
```

❖ getElementByName:

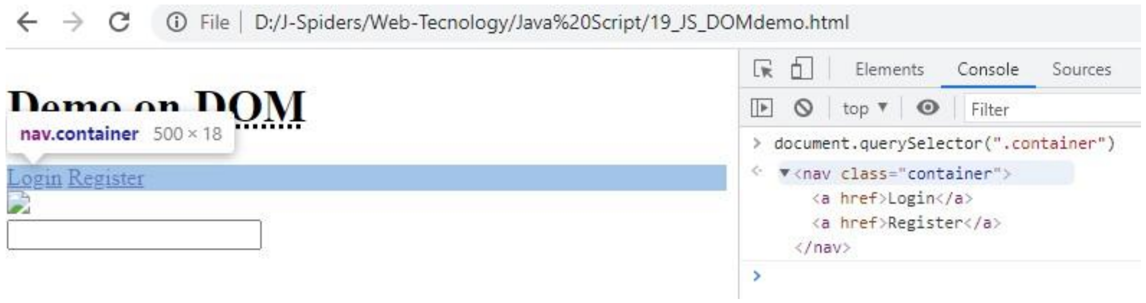
- This method helps to select HTML elements based on Name Attribute.
- This method takes name as Arguments in the form of String.
- This method will return all the selected HTML elements in the form of array



```
> document.getElementsByName('username')[0]
<· <input type="text" name="username">
```

❖ querySelector:

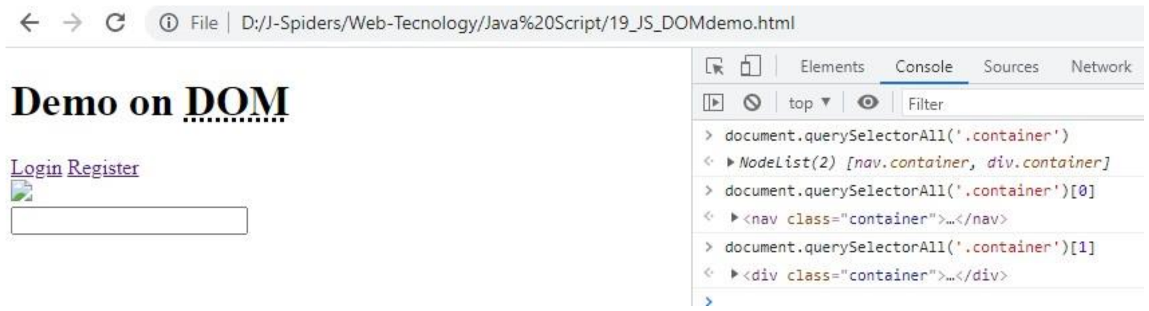
- This method helps to select HTML elements based on CSS Selectors.
- We can pass class selector(.classname), id selector(#idname), Element Selector(Tagname), Attribute Selector etc…
- This method will return Only the 1st match



```
> document.querySelector(".container")
<· ▼<nav class="container">
      <a href>Login</a>
      <a href>Register</a>
    </nav>
```

❖ querySelectorAll:

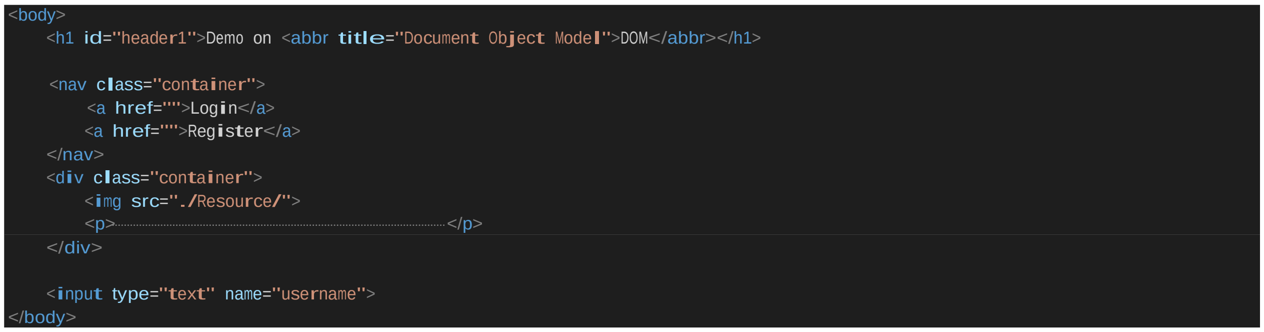
- This method is same that as query selector but this method will return all the matches in the form of array



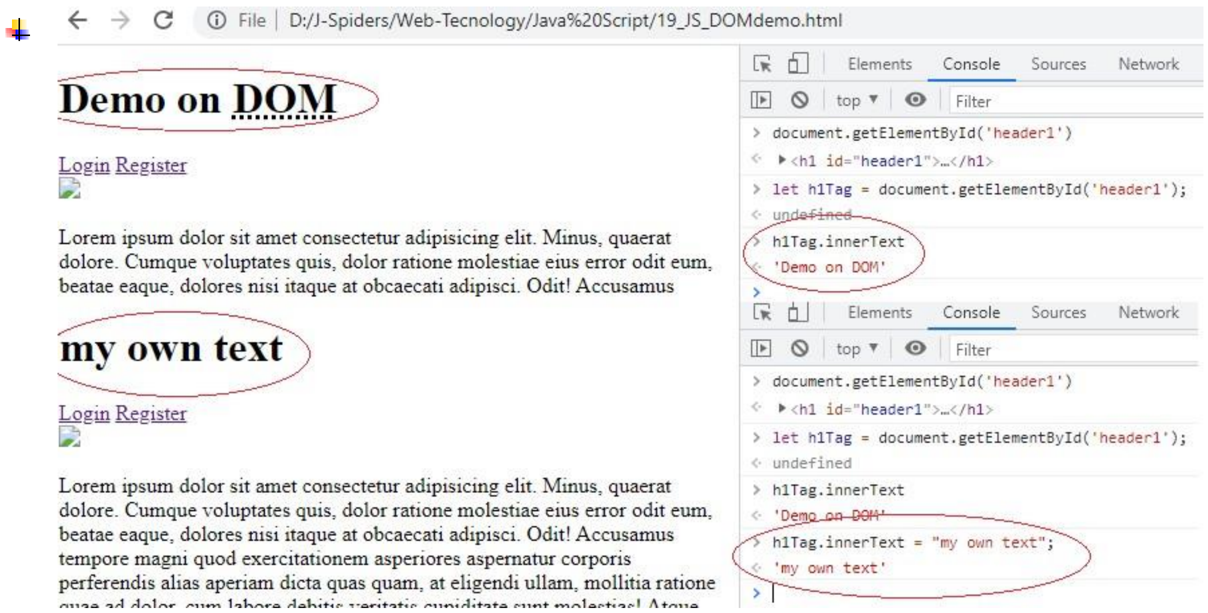
```
> document.querySelectorAll('.container')
<· ▶NodeList(2) [nav.container, div.container]
> document.querySelectorAll('.container')[0]
<· ▶<nav class="container">…</nav>
> document.querySelectorAll('.container')[1]
<· ▶<div class="container">…</div>
```

<u>Manipulation:</u>

- ✓ Changing the contents
  selectedElement.innerText.
  selectedElement.innerHTML.



```
<body>
    <h1 id="header1">Demo on <abbr title="Document Object Model">DOM</abbr></h1>

    <nav class="container">
        <a href="">Login</a>
        <a href="">Register</a>
    </nav>
    <div class="container">
        <img src="./Resource/">
        <p>..............................................................</p>
    </div>

    <input type="text" name="username">
</body>
```



File | D:/J-Spiders/Web-Tecnology/Java%20Script/19_JS_DOMdemo.html

Demo on DOM

Login Register

Lorem ipsum dolor sit amet consectetur adipisicing elit. Minus, quaerat dolore. Cumque voluptates quis, dolor ratione molestiae eius error odit eum, beatae eaque, dolores nisi itaque at obcaecati adipisci. Odit! Accusamus

```
> document.getElementById('header1')
<· ▸<h1 id="header1">…</h1>
> let h1Tag = document.getElementById('header1');
<· undefined
> h1Tag.innerText
<· 'Demo on DOM'
>
```

my own text

Login Register

Lorem ipsum dolor sit amet consectetur adipisicing elit. Minus, quaerat dolore. Cumque voluptates quis, dolor ratione molestiae eius error odit eum, beatae eaque, dolores nisi itaque at obcaecati adipisci. Odit! Accusamus tempore magni quod exercitationem asperiores aspernatur corporis perferendis alias aperiam dicta quas quam, at eligendi ullam, mollitia ratione

```
> document.getElementById('header1')
<· ▸<h1 id="header1">…</h1>
> let h1Tag = document.getElementById('header1');
<· undefined
> h1Tag.innerText
<· 'Demo on DOM'
> h1Tag.innerText = "my own text";
<· 'my own text'
> |
```