SYLLABUS

Dynamic HTML (Cont..): JavaScript Objects, Working with Browser Objects, Working with Document Object.
Document Object Model: Understanding DOM Nodes, Understanding DOM Levels,
Understanding DOM Interfaces - Node , Document, Element, Attribute.

Dynamic HTML(Cont..): JavaScript Objects

Exploring Objects in JavaScript:

- As we already know, JavaScript language is an object-based language.
- Object is real world entity of instance.
- Every object having properties and methods.

Properties of an Object:

- Property (attribute) is the characteristic of an object.
- Properties are added by using this keyword followed by dot (.) symbol.
- For example, student is an object then name and rollno are properties.

Methods of an object:

- Method is a set of one or more statements that are executed by referring the name of the method.
- To add methods to a user defined object, you need to perform following steps:
  - ✓ Declaring and defining a function for each method.
  - ✓ Associating a function with an object.

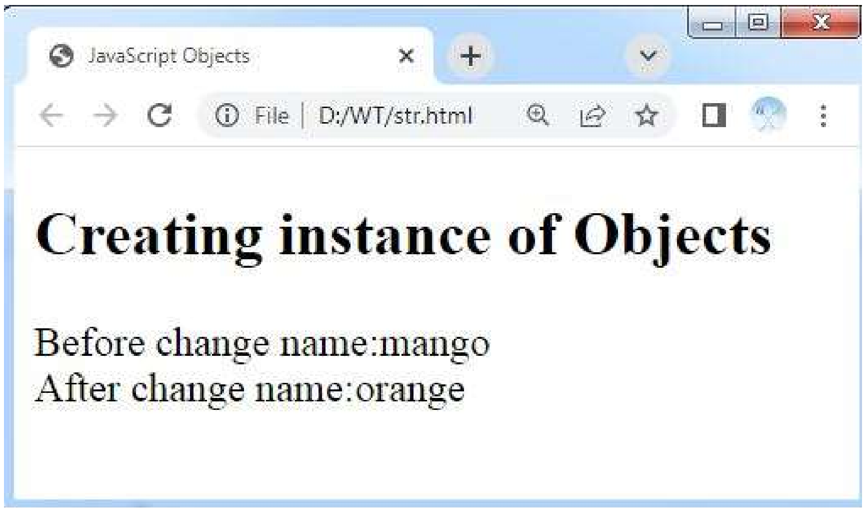
- In JavaScript, an object is created in two ways: **direct instance** and **function template**

1. **<u>direct instance:</u>**
    - Direct instance is created by using ***new*** keyword.
    - Syntax: **obj = new object();**
    - Example: **student = new object();**
    - You can also add properties and methods to object by using dot (.) symbol.
    - Example: **student.name = ”apple”;**
      student.rollnum = 5;
      student.getmarks();

Example: Write a JS program to demonstrate the instance object

```
<!DOCTYPE html>
<html>
        <head>
                <title>JavaScript Objects</title>
        </head>
        <body >
                <script type="text/javascript">
                        document.write("<h2>Creating instance of Objects </h2>");
                        var s = new Object();
                        s.name="mango";
                        s.display=function (n) {
                                s.name=n;
                        }
                        document.write("Before change name:"+s.name);
                        s.display("orange");
                        document.write("<br>After change name:"+s.name);
                </script>
        </body>
</html>
```

<u>Output:</u>



```
Creating instance of Objects

Before change name:mango
After change name:orange
```

2. **function template:**

- First you need to define function template by using ***function*** keyword.
- After defining the function template for an object, you need to create an instance of the object by using ***new*** keyword.
- You can also add properties to function template by using ***this*** keyword.
- Example: define function

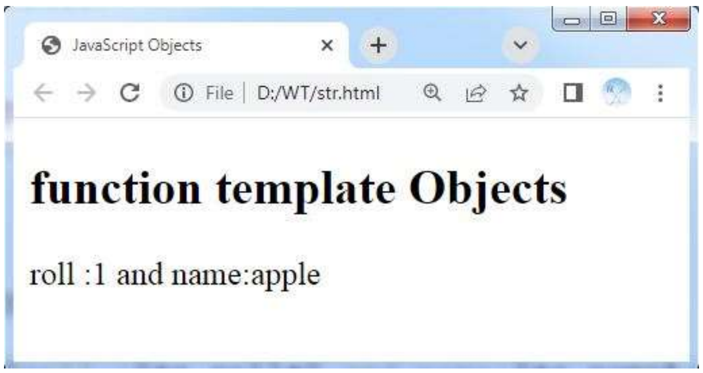
```
function        vehicle(name, color)
{
        this . name = name;
        this . color = color;
}
```

- Create object: var   bike = new vehicle("shine" , "black");
- Access the properties: var   name_of_bike = bike.name;

Example: Write a JS program to demonstrate the instance object

```
<!DOCTYPE html>
<html>
        <head>
                <title>JavaScript Objects</title>
        </head>
        <body >
                <script type="text/javascript">
                        document.write("<h2>function template Objects </h2>");
                        function student(n,r){
                                this.name=n;
                                this.roll=r;
                                this.display=display;
                                function display(){
                                        document.write("roll :"+s.roll+" and name:"+s.name);
                                }
                        }
                        var s = new student("apple",1);
                        s.display();
                </script>
        </body>
</html>
```

Output:



function template Objects

roll :1 and name:apple

Exploring Standard/Built-in JavaScript Objects:

- JavaScript support seven built-in objects, they are

S.No	Object Name
1	String object
2	Number object
3	Math object
4	Date object
5	Array object
6	Boolean
7	RegExp

- These objects also called as core language objects.

1. String Objects:

- String is a series of characters.
- String objects are used to perform operations on text.

Syntax:

var variable_name = new String(string);

Or

var variable_name=”string”;

- **Example:**

var s = new String(“Hello”);

Or

var s=”Hello”;

- **String object properties**

Properties	Description
Length	It returns the length of the string.
Prototype	It allows you to add properties and methods to an object.
Constructor	It returns the reference to the String function that created the object.

- **String object methods**

Methods	Description
charAt()	It returns the character at the specified index.
charCodeAt()	It returns the ASCII code of the character at the specified position.
concat()	It combines the text of two strings and returns a new string.
indexOf()	It returns the index within the calling String object.
lastIndesOf()	It returns the position of last occurrence of specified value in string.
match()	It is used to match a regular expression against a string.
quote()	It writes the quotes in JavaScript
replace()	It is used to replace the matched substring with a new substring.
search()	It executes the search for a match between regular expressions.
slice()	It extracts a session of a string and returns a new string.
split()	It splits a string into substrings.
toLowerCase()	It returns the calling string value converted lower case.
toUpperCase()	Returns the calling string value converted to uppercase.

fromCharCode()	Converts Unicode to character.
substring()	It returns the string between two specified indices.
toSource()	It returns an object literal representing the specified object.
toString()	It returns string representing the specified object.
valueOf()	It returns the primitive value of a String object

- **Wrapper methods of String Object:**

Method	**Description**
anchor()	Create an anchor
big()	Display the string with big font
blink()	Make the string to blink
bold()	Make the string bold
fixed()	Display the string by using fixed pitch font
fontcolor()	Display the string with specified color
fontsize()	Display the string with specified size
italics()	Display the string the string in italic
link()	Display the string as hyper link
small()	Display the string with small font
strike()	Display the string with a strike-out text
sub()	Display the string as sub script
sup()	Display the string as super script

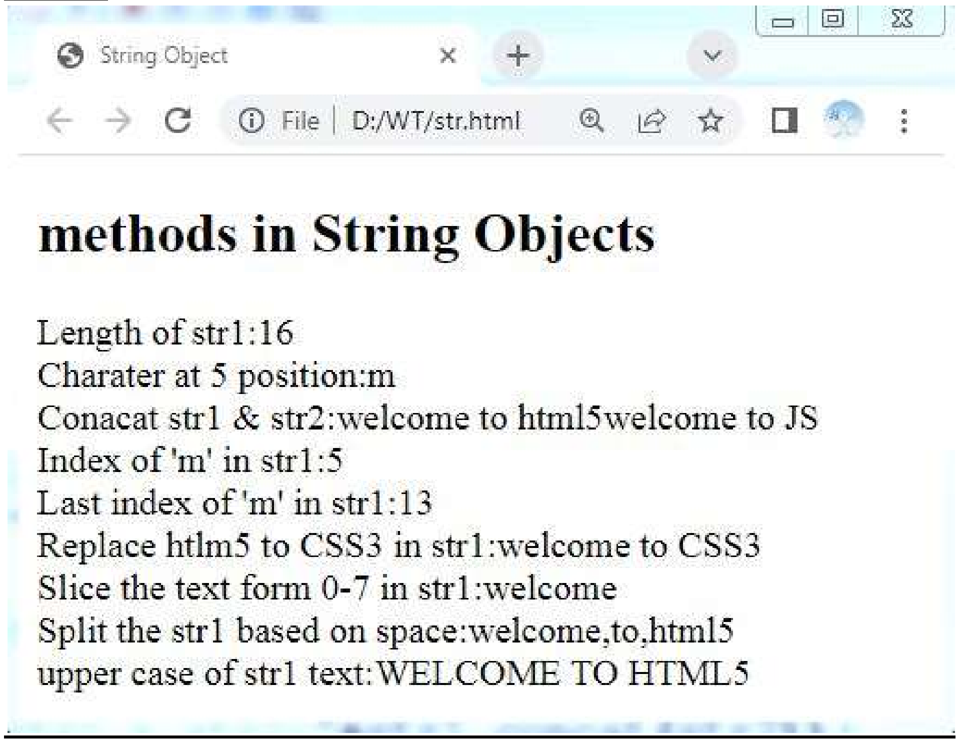
- **Comparison operators**

Operator	**Description**
!=	Represents the not equal operation
<	Represents the less than operation
>	Represents the greater than operation
+	Represents the concatenate operation
<=	Represents the less than or equal to operation
>=	Represents the greater than or equal to operation
+=	Represents the concatenate assignment operation
==	Represents the equal operation

Example1: Write a JS to demonstrate the methods in String objects

```
<!DOCTYPE html>
<html>
<head>
        <title>String Object</title>
</head>
<body >
        <script type="text/javascript">
                document.write("Properties and methods in String Objects");
                var str1 = "welcome to html5";
                var str2 = "welcome to JS";
                document.write("Length of str1:"+str1.length);
                document.write("<br>Charater at 5 position:"+str1.charAt(5));
                document.write("<br>Conacat str1 & str2:"+str1.concat(str2));
                document.write("<br>Index of 'm' in str1:"+str1.indexOf('m'));
                document.write("<br>Last index of 'm' in str1:"+str1.lastIndexOf('m'));
        document.write("<br>Replace htlm5 to CSS3 in str1:"+str1.replace("html5","CSS3"));
                document.write("<br>Slice the text form 0-7 in str1:"+str1.slice(0,7));
                document.write("<br>Split the str1 based on space:"+str1.split(" "));
                document.write("<br>upper case of str1 text:"+str1.toUpperCase());
        </script>
</body>
</html>
```

Output:



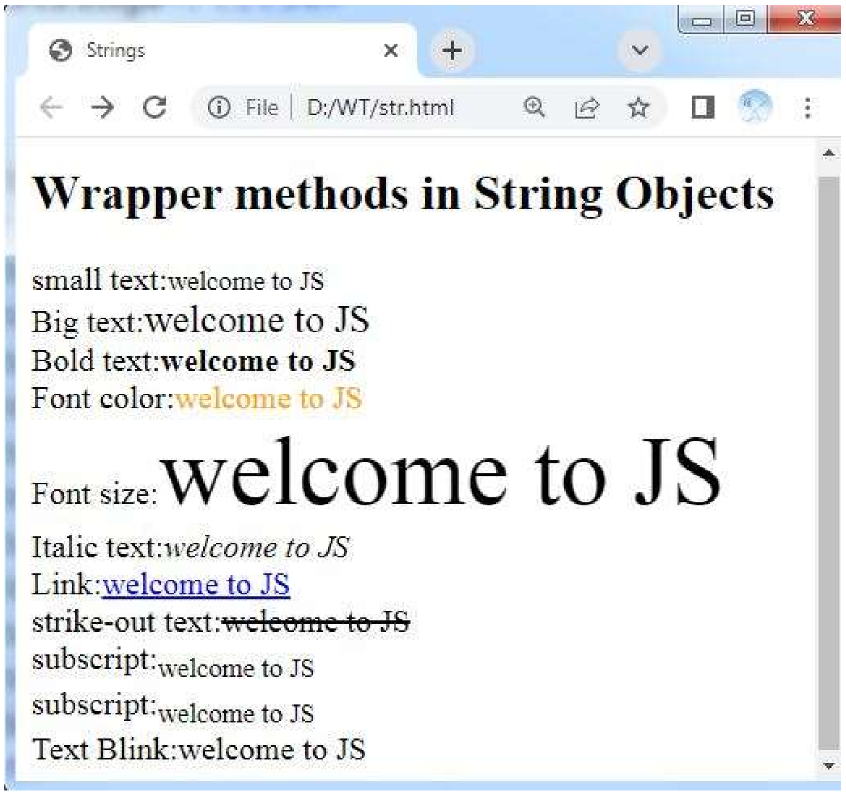
```
methods in String Objects

Length of str1:16
Charater at 5 position:m
Conacat str1 & str2:welcome to html5welcome to JS
Index of 'm' in str1:5
Last index of 'm' in str1:13
Replace htlm5 to CSS3 in str1:welcome to CSS3
Slice the text form 0-7 in str1:welcome
Split the str1 based on space:welcome,to,html5
upper case of str1 text:WELCOME TO HTML5
```

Example2: Write a JS to demonstrate the wrapper methods in String objects

```
<!DOCTYPE html>
<html>
<head>
        <title>String</title>
</head>
<body >
        <script type="text/javascript">
                document.write("<h2>Wrapper methods in String Objects </h2>");
                var str = "welcome to JS";
                document.write("small text:"+str.small());
                document.write("<br>Big text:"+str.big());
                document.write("<br>Bold text:"+str.bold());
                document.write("<br>Font color:"+str.fontcolor("orange"));
                document.write("<br>Font size:"+str.fontsize("10px"));
                document.write("<br>Italic text:"+str.italics());
                document.write("<br>Link:"+str.link());
                document.write("<br>strike-out text:"+str.strike());
                document.write("<br>subscript:"+str.sub());
                document.write("<br>subscript:"+str.sub());
                document.write("<br>Text Blink:"+str.blink());
        </script>
</body>
</html>
```

Output:



Wrapper methods in String Objects

small text:welcome to JS
Big text:welcome to JS
Bold text:**welcome to JS**
Font color:welcome to JS
Font size:welcome to JS
Italic text:*welcome to JS*
Link:welcome to JS
strike-out text:~~welcome to JS~~
subscript:welcome to JS
subscript:welcome to JS
Text Blink:welcome to JS

2. Number Object:

- All numbers in JavaScript are 64 bit(b bytes) floating point numbers and the range is 5e-324(negative) to 1.7976931348623157e+308(positive)
- Syntax: var num = new Number(value)
- Example: var n = new Number('21.5768');
- **Properties:**

Property	**Description**
MIN_VALUE	Returns the minimum numerical value possible in JavaScript
MAX_VALUE	Returns the maximum numerical value possible in JavaScript
NEGATIVE_INFINITY	Represents the value of negative infinity
POSITIVE_INFINITY	Represents the value of infinity
Constructor	Holds the value of constructor function that has created the object.
Prototype	Adds property and methods to Number object

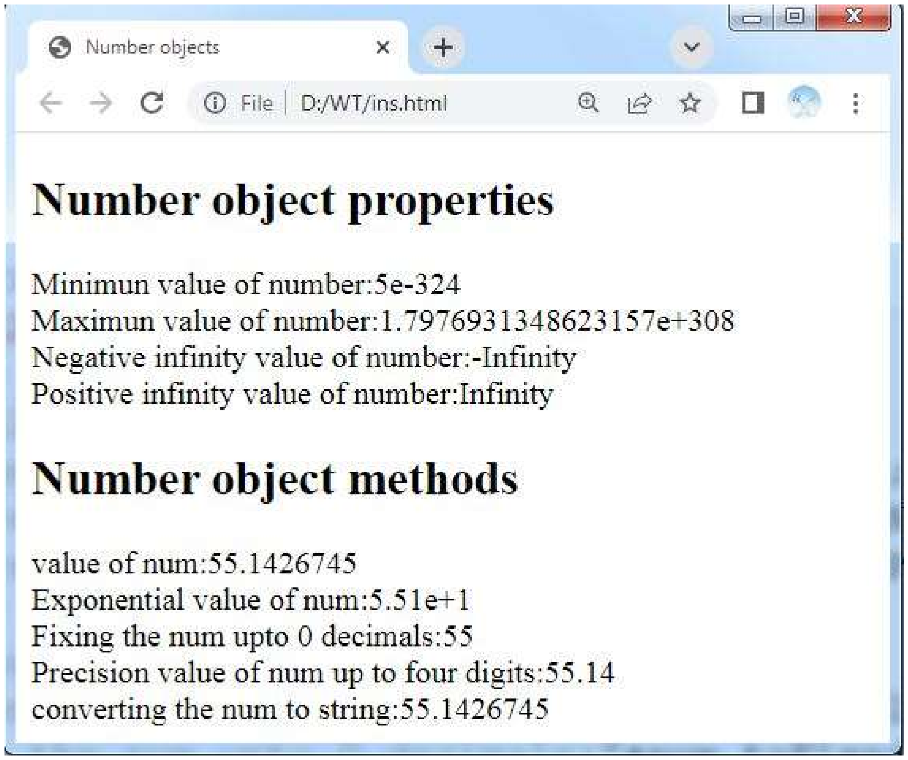
- **Methods:**

Property	**Description**
toExponential(x)	Converts a number into an exponential notation.
toFixed(x)	Round up a number to x digits after the decimal
toPrecision(x)	Round up a number to a length of x digits
toString(x)	Returns a string value for the Number object
valueOf()	Returns a primitive value for the Number type.

Example: Write a JS to demonstrate the properties and methods in Number objects

```
<!DOCTYPE html>
<html>
    <head>
            <title>Number objects</title>
    </head>
    <body >
            <script type="text/javascript">
                    var num = new Number(55.1426745);
                    document.write("<h2>Number object properties</h2>");
                    document.write("Minimun value of number:"+Number.MIN_VALUE);
                    document.write("<br>Maximun value of number:"+Number.MAX_VALUE);
    document.write("<br>Negative infinity value of number:"+Number.NEGATIVE_INFINITY);
    document.write("<br>Positive infinity value of number:"+Number.POSITIVE_INFINITY);
                    document.write("<h2>Number object methods</h2>");
                    document.write("value of num:"+num.valueOf());
                    document.write("<br>Exponential value of num:"+num.toExponential(2));
                    document.write("<br>Fixing the num upto 0 decimals:"+num.toFixed(0));
                    document.write("<br>Precision value of num up to four digits:"+num.toPrecision(4));
                    document.write("<br>converting the num to string:"+num.toString());
            </script>
    </body>
</html>
```

Output:



```
Number object properties

Minimun value of number:5e-324
Maximun value of number:1.7976931348623157e+308
Negative infinity value of number:-Infinity
Positive infinity value of number:Infinity

Number object methods

value of num:55.1426745
Exponential value of num:5.51e+1
Fixing the num upto 0 decimals:55
Precision value of num up to four digits:55.14
converting the num to string:55.1426745
```

3. Math Object:

- The math object is used to perform simple and complex arithmetic operation.
- JavaScript math object is used to perform mathematical task.
- Properties:

Property	Description	
	Holds	approximate value
E	Euler's number	2.718
LN2	natural logarithmic of 2	0.693
LN10	natural logarithmic of 10	2.302
LOG2E	base-2 logarithmic of E	1.442
LOG10E	base-10 logarithmic of E	0.434
PI	Number value of PI	3.142
SQRT1_2	Square root of 1/2	0.707
SQRT2	Square root of 2	1.414

- Methods

Method	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x, in radians
asin(x)	Returns the arcsine of x, in radians
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y, x)	Returns the arctangent of the quotient of its arguments
ceil(x)	Returns x, rounded upwards to the nearest integer
cos(x)	Returns the cosine of x (x is in radians)
exp(x)	Returns the value of Ex
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm of x
max(x1,x2,..xn)	Returns the number with the highest value
min(x1,x2,..xn)	Returns the number with the lowest value
pow(x, y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sin(x)	Returns the sine of x (x is in radians)
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle

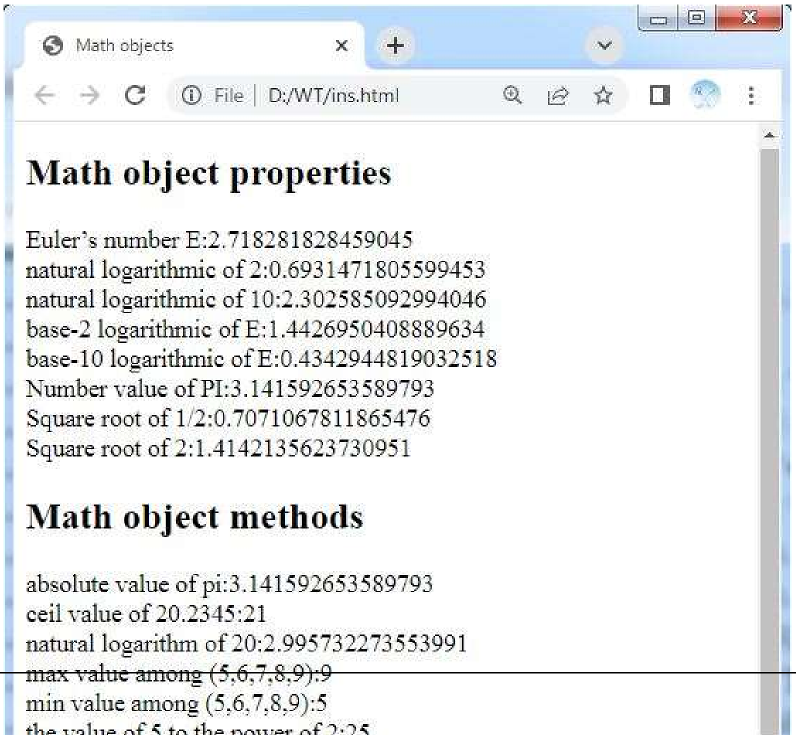
Example: Write a JS to demonstrate the properties and methods in Math objects

```
<!DOCTYPE html>
<html>
    <head>
            <title>Math objects</title>
    </head>
    <body >
            <script type="text/javascript">
                    document.write("<h2>Math object properties</h2>");
                    document.write("Euler’s number E:"+Math.E);
                    document.write("<br>natural logarithmic of 2:"+Math.LN2);
                    document.write("<br>natural logarithmic of 10:"+Math.LN10);
                    document.write("<br>base-2 logarithmic of E:"+Math.LOG2E);
                    document.write("<br>base-10 logarithmic of E:"+Math.LOG10E);
                    document.write("<br>Number value of PI:"+Math.PI);
                    document.write("<br>Square root of 1/2:"+Math.SQRT1_2);
                    document.write("<br>Square root of 2:"+Math.SQRT2);


            document.write("<h2>Math object methods</h2>");

                    document.write("absolute value of pi:"+Math.abs(Math.PI));
                    document.write("<br>ceil value of 20.2345:"+Math.ceil(20.2345));
                    document.write("<br>natural logarithm of 20:"+Math.log(20));
                    document.write("<br>max value among (5,6,7,8,9):"+Math.max(5,6,7,8,9));
                    document.write("<br>min value among (5,6,7,8,9):"+Math.min(5,6,7,8,9));
                    document.write("<br>the value of 5 to the power of 2:"+Math.pow(5,2));
                    document.write("<br>random number between 0 and 1:"+Math.random());
                    document.write("<br>square root of 16:"+Math.sqrt(16));
            </script>
    </body>
</html>
```

Output:



Math object properties

```
Euler’s number E:2.718281828459045
natural logarithmic of 2:0.6931471805599453
natural logarithmic of 10:2.302585092994046
base-2 logarithmic of E:1.4426950408889634
base-10 logarithmic of E:0.4342944819032518
Number value of PI:3.141592653589793
Square root of 1/2:0.7071067811865476
Square root of 2:1.4142135623730951
```

Math object methods

```
absolute value of pi:3.141592653589793
ceil value of 20.2345:21
natural logarithm of 20:2.995732273553991
max value among (5,6,7,8,9):9
min value among (5,6,7,8,9):5
the value of 5 to the power of 2:25
```

4. Date Object:

- The Date object used to display the date on web page or time stamp in numerical or mathematical calculations.
- Date objects are created with new Date ().
- There are five ways of instantiating (creating) a date:
  var date1 = new Date();
  or
  var date1 = new Date(*milliseconds*);
  or
  var date1 = new Date(*"mm dd, yyyy"*);
  or
  var date1 = new Date(*"mm dd, yyyy hr:min:sec"*);
  or
  var date1 = new Date(*yyyy*, *mm*, *dd*, [, *hr*, *min*, *sec*, *millisec]*);

- properties:

Name	**Description**
Constructor	Returns the function that created the Date object's prototype
Prototype	Allows you to add properties and methods to an object

- Methods

Name	**Description**
getDate()	Returns the day of the month (from 1-31)
getDay()	Returns the day of the week (from 0-6)
getFullYear()	Returns the year
getHours()	Returns the hour (from 0-23)
getMilliseconds()	Returns the milliseconds (from 0-999)
getMinutes()	Returns the minutes (from 0-59)
getMonth()	Returns the month (from 0-11)
getSeconds()	Returns the seconds (from 0-59)
getTime()	Returns the number of milliseconds since midnight Jan 1 1970, and a specified date
getTimezoneOffset()	Returns the time difference between UTC time and local time, in minutes
getUTCDate()	Returns the day of the month, according to universal time (from 1-31)
getUTCDay()	Returns the day of the week, according to universal time (from 0-6)
getUTCFullYear()	Returns the year, according to universal time
getUTCHours()	Returns the hour, according to universal time (from 0-23)
getUTCMilliseconds()	Returns the milliseconds, according to universal time (from 0-999)
getUTCMinutes()	Returns the minutes, according to universal time (from 0-59)
getUTCMonth()	Returns the month, according to universal time (from 0-11)
getUTCSeconds()	Returns the seconds, according to universal time (from 0-59)
now()	Returns the number of milliseconds since midnight Jan 1, 1970
parse()	Parses a date string and returns the number of milliseconds since January 1, 1970

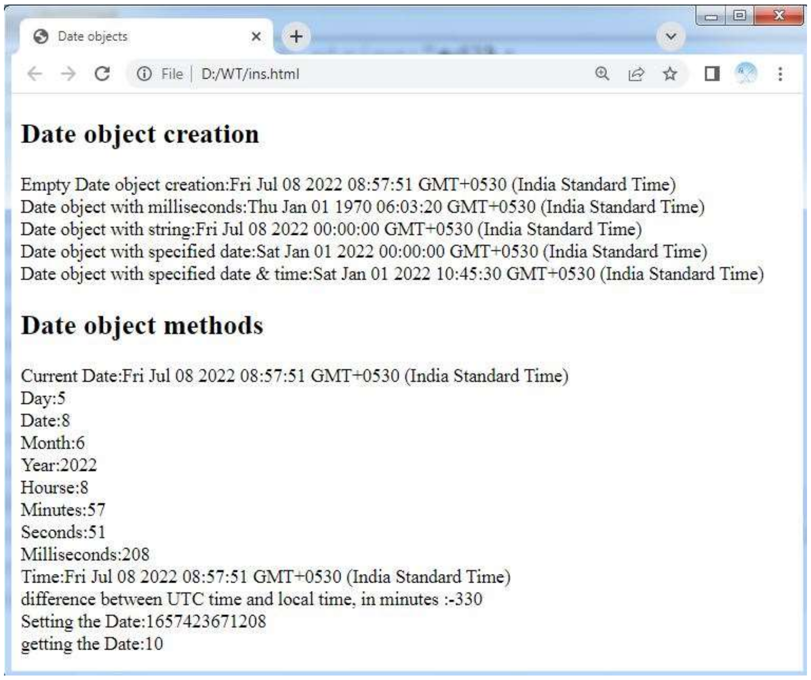
setDate()	Sets the day of the month of a date object
setFullYear()	Sets the year of a date object
setHours()	Sets the hour of a date object
setMilliseconds()	Sets the milliseconds of a date object
setMinutes()	Set the minutes of a date object
setMonth()	Sets the month of a date object
setSeconds()	Sets the seconds of a date object
setTime()	Sets a date to a specified number of milliseconds after/before January 1, 1970
setUTCDate()	Sets the day of the month of a date object, according to universal time
setUTCFullYear()	Sets the year of a date object, according to universal time
setUTCHours()	Sets the hour of a date object, according to universal time
setUTCMilliseconds()	Sets the milliseconds of a date object, according to universal time
setUTCMinutes()	Set the minutes of a date object, according to universal time
setUTCMonth()	Sets the month of a date object, according to universal time
setUTCSeconds()	Set the seconds of a date object, according to universal time
setYear()	Deprecated. Use the setFullYear() method instead
toDateString()	Converts the date portion of a Date object into a readable string
toLocaleDateString()	Returns the date portion of a Date object as a string, using locale conventions
toLocaleTimeString()	Returns the time portion of a Date object as a string, using locale conventions
toLocaleString()	Converts a Date object to a string, using locale conventions
toString()	Converts a Date object to a string
toTimeString()	Converts the time portion of a Date object to a string
toUTCString()	Converts a Date object to a string, according to universal time
UTC()	Returns the number of milliseconds in a date since midnight of January 1, 1970, according to UTC time
valueOf()	Returns the primitive value of a Date object

Example: Write a JS to demonstrate the creation and methods of Date objects

```
<!DOCTYPE html>
<html>
        <head>
                <title>Date objects</title>
        </head>
        <body >
                <script type="text/javascript">
                        var d1 = new Date();
                        var d2 = new Date(2000000);
                        var d3 = new Date("july 08,2022");
                        var d4 = new Date(2022,00,01);
                        var d5 = new Date(2022,00,01,10,45,30);
                        document.write("<h2>Date object creation</h2>");
                        document.write("Empty Date object creation:"+d1);
                        document.write("<br>Date object with milliseconds:"+d2);
                        document.write("<br>Date object with string:"+d3);
                        document.write("<br>Date object with specified date:"+d4);
                        document.write("<br>Date object with specified date & time:"+d5);

                        document.write("<h2>Date object methods</h2>");
                        document.write("Current Date:"+Date(d1.valueOf()));
                        document.write("<br>Day:"+d1.getDay());
                        document.write("<br>Date:"+d1.getDate());
                        document.write("<br>Month:"+d1.getMonth());
                        document.write("<br>Year:"+d1.getFullYear());
                        document.write("<br>Hourse:"+d1.getHours());
                        document.write("<br>Minutes:"+d1.getMinutes());
                        document.write("<br>Seconds:"+d1.getSeconds());
                        document.write("<br>Milliseconds:"+d1.getMilliseconds());
                        document.write("<br>Time:"+Date(d1.getTime()));
document.write("<br>difference between UTC time and local time, in minutes :"+d1.getTimezoneOffset());
                        document.write("<br>Setting the Date:"+d1.setDate(10));
                        document.write("<br>getting the Date:"+d1.getDate());
                </script>
        </body>
</html>
```

Output:



Date object creation

```
Empty Date object creation:Fri Jul 08 2022 08:57:51 GMT+0530 (India Standard Time)
Date object with milliseconds:Thu Jan 01 1970 06:03:20 GMT+0530 (India Standard Time)
Date object with string:Fri Jul 08 2022 00:00:00 GMT+0530 (India Standard Time)
Date object with specified date:Sat Jan 01 2022 00:00:00 GMT+0530 (India Standard Time)
Date object with specified date & time:Sat Jan 01 2022 10:45:30 GMT+0530 (India Standard Time)
```

Date object methods

```
Current Date:Fri Jul 08 2022 08:57:51 GMT+0530 (India Standard Time)
Day:5
Date:8
Month:6
Year:2022
Hourse:8
Minutes:57
Seconds:51
Milliseconds:208
Time:Fri Jul 08 2022 08:57:51 GMT+0530 (India Standard Time)
difference between UTC time and local time, in minutes :-330
Setting the Date:1657423671208
getting the Date:10
```

5. Array object:

- An **Array** is used to store a number of values (called as elements) in order with a single variable.
- An array object can created in two ways

Using array constructor:

- Creates an empty array: var ar = new Array()
- Creates an array with given size: var ar = new Array(size)
- Creates an array based on number of elements:
  var ar = new Array(element0, element1, ..., elementN)

Using array literal notation:

- Array literal notation are coma separated list of items enclosed by square brackets.
- creates an empty array: var ar = [ ]
- creates an empty array with elements: var ar = [ele1,ele2,...,elen]
- example: var ar = [5,"hello", true]

- **Javascript Array Objects Property**

Name	**Description**
Length	Returns the number of elements in an array
prototype	Use to add new properties to all objects.
constructor	Specifies the function which creates an object's prototype.

- **JavaScript object methods:**

Name	**Description**
concat()	Use to join two or more arrays and returns a new array.
join()	Use to join all elements of an array into a string.
pop()	Use to remove the last element from an array.
push()	Use to add one or more elements at the end of an array.
reverse()	Use to reverse the order of the elements in an array.
shift()	Use to remove first element from an array.
slice()	Use to extract a section of an array.
sort()	Use to sort the elements of an array.
toString()	Returns a string represent the array and its elements.
unshift()	Use to add one or more elements to the beginning of an array and returns the new length of the array.
valueOf()	Returns the primitive value of an array.

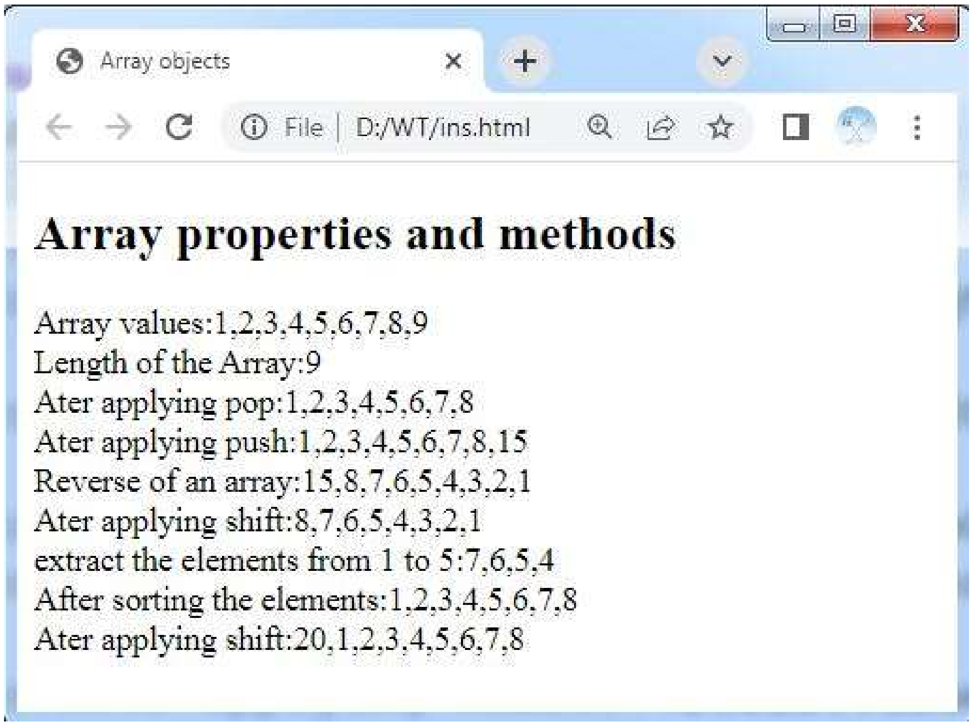
Example: Write a JS to demonstrate the properties and methods of Array objects

```
<!DOCTYPE html>
<html>
        <head>
                <title>Array objects</title>
        </head>
        <body >
                <script type="text/javascript">
                        var  rolls = new Array(1,2,3,4,5,6,7,8,9);

                        document.write("<h2>Array properties and methods</h2>");

                        document.write("Array values:"+rolls.valueOf());
                        document.write("<br>Length of the Array:"+rolls.length);
                        document.write("Popped element is”+rolls.pop());
                        document.write("<br>Ater applying pop:"+rolls.valueOf());
                        document.write(“pushed element”+rolls.push(15));
                        document.write("<br>Ater applying push:"+rolls.valueOf());
                        document.write("<br>Reverse of an array:"+rolls.reverse());
                        document.write(“”+rolls.shift());
                        document.write("<br>Ater applying shift:"+rolls.valueOf());
                        document.write("<br>extract the elements from 1 to 5:"+rolls.slice(1,5));
                        rolls.sort();
                        document.write("<br>After sorting the elements:"+rolls.sort());
                        rolls.unshift(20);
                        document.write("<br>Ater applying shift:"+rolls.valueOf());
                </script>
        </body>
</html>
```

Output:



```
Array properties and methods

Array values:1,2,3,4,5,6,7,8,9
Length of the Array:9
Ater applying pop:1,2,3,4,5,6,7,8
Ater applying push:1,2,3,4,5,6,7,8,15
Reverse of an array:15,8,7,6,5,4,3,2,1
Ater applying shift:8,7,6,5,4,3,2,1
extract the elements from 1 to 5:7,6,5,4
After sorting the elements:1,2,3,4,5,6,7,8
Ater applying shift:20,1,2,3,4,5,6,7,8
```

6. Boolean object:

- A Boolean object is an object which holds Boolean values.
- Boolean primitive value (true and false) is not same as Boolean object values (true and false).
- It is used to converts non Boolean values into Boolean values.
- Boolean object can created in three ways
- Using Boolean literals notation
- Example: var bool = true;

- Using Boolean object constructor
- Example:
```
var bool = Boolean(true);
var num = 10;
val bool = Boolean(num>5)
```
- Using testable expression
- Testable expressions are used to evaluate the output in Boolean values.
- Example:
```
if(5>2){ ... }
if(true){ ... }
```

- **JavaScript Boolean Objects Property**

Name	**Description**
constructor	Specifies the function that creates an object's prototype.
prototype	Use to add new properties and methods to a Boolean object.

- **JavaScript Boolean Objects Methods**

Name	**Description**
toString	Returns a string representing the specified boolean object.
valueof	Returns the primitive value of a boolean object.

7. RegExp Object:

- A regula expression (RegExp) is an object that validates the pattern of characters.
- The pattern is used to do pattern-matching "search-and-replace" functions on text.
- In JavaScript, a RegExp Object is a pattern with Properties and Methods.
- The following ways are creating the object
- Using RegExp object constructor
- Syntax: var pattern = new RegExp("pattern", "flag");
- Example: match the pattern xxx.xxx.xxx.xxx
```
var pattern = new RegExp("\\b\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\b", "g");
```
- Using literals
- Syntax: var pattern = pattern/flag;
- Example: match the pattern xxx.xxx.xxx.xxx
```
var pattern = /\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b/g;
```

- Modifiers (flag): Modifiers are used to perform case-insensitive and global searches:

Modifier	**Description**
G	Perform a global match (find all matches rather than stopping after the first match)
I	Perform case-insensitive matching
M	Perform multiline matching

- Regular expression with range of characters

Expression	**Description**
[abc]	Find any character between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find any character between the brackets (any digit)
[^0-9]	Find any character NOT between the brackets (any non-digit)
(x|y)	Find any of the alternatives specified

- **Meta characters:** Meta characters are characters with a special meaning:

Metacharacter	**Description**
.	Find a single character, except newline or line terminator
\w	Find a word character
\W	Find a non-word character
\d	Find a digit
\D	Find a non-digit character
\s	Find a whitespace character
\S	Find a non-whitespace character
\b	Find a match at the beginning/end of a word, beginning like this: \bHI, end like this: HI\b
\B	Find a match, but not at the beginning/end of a word
\0	Find a NULL character
\n	Find a new line character
\f	Find a form feed character
\r	Find a carriage return character
\t	Find a tab character
\v	Find a vertical tab character
\xxx	Find the character specified by an octal number xxx
\xdd	Find the character specified by a hexadecimal number dd
\udddd	Find the Unicode character specified by a hexadecimal number dddd

- Quantifiers

Quantifier	**Description**
n+	Matches any string that contains at least one *n*
n*	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*
n{X}	Matches any string that contains a sequence of *X n*'s
n{X,Y}	Matches any string that contains a sequence of X to Y *n*'s
n{X,}	Matches any string that contains a sequence of at least X *n*'s
n$	Matches any string with *n* at the end of it
^n	Matches any string with *n* at the beginning of it
?=n	Matches any string that is followed by a specific string *n*
?!n	Matches any string that is not followed by a specific string *n*

- RegExp Object Properties

Property	**Description**
Constructor	Returns the function that created the RegExp object's prototype
Global	Checks whether the "g" modifier is set
ignoreCase	Checks whether the "i" modifier is set
lastIndex	Specifies the index at which to start the next match
Multiline	Checks whether the "m" modifier is set
Source	Returns the text of the RegExp pattern

- RegExp Object Methods

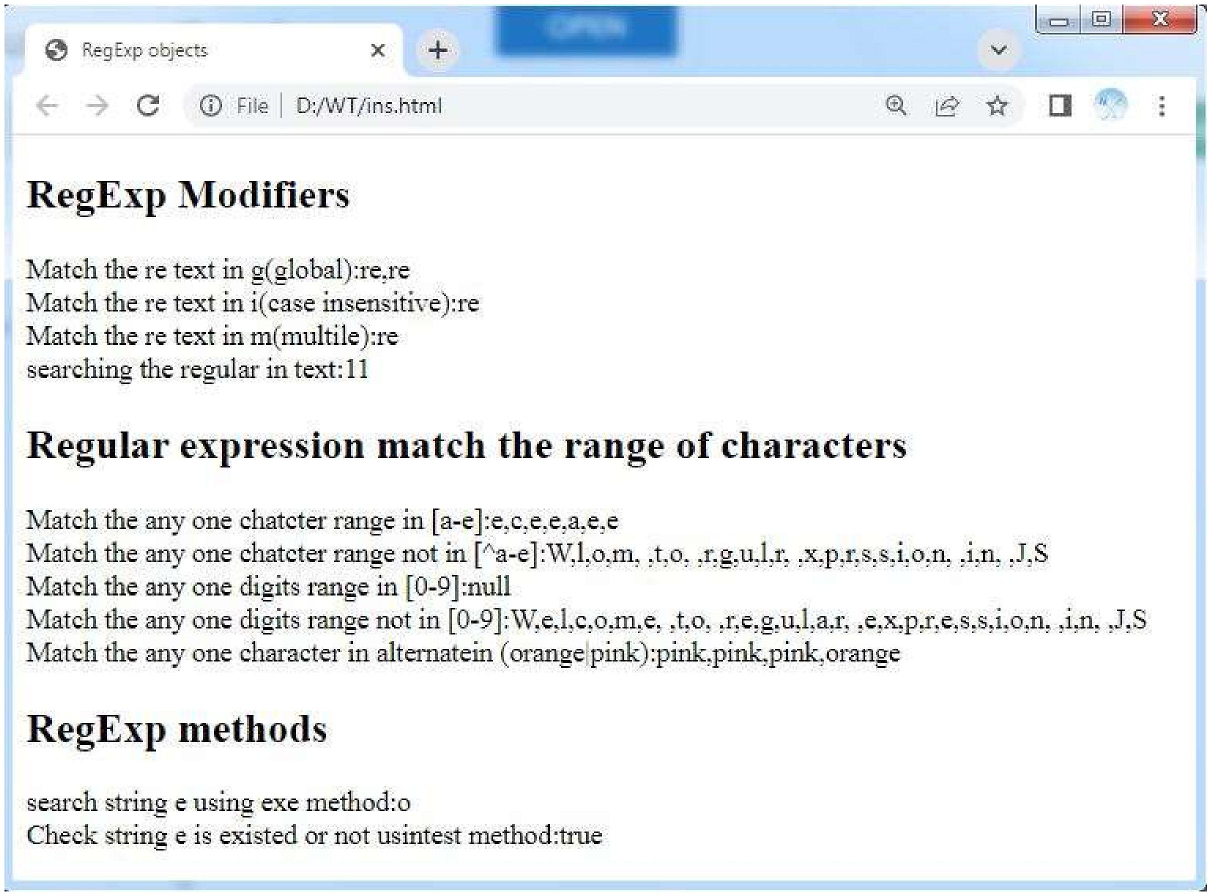
Method	**Description**
compile()	Deprecated in version 1.5. Compiles a regular expression
exec()	Tests for a match in a string. Returns the first match
test()	Tests for a match in a string. Returns true or false
toString()	Returns the string value of the regular expression

Example: Write a JS to demonstrate the properties and methods of Array objects

```
<!DOCTYPE html>
<html>
        <head>
                <title>RegExp objects</title>
        </head>
        <body >
                <script type="text/javascript">
                        var text ="Welcome to regular expression in JS";
                        var colors="pink, pink, yellow, pink, blue, orange, red"
                        document.write("<h2>RegExp Modifiers</h2>");
                        document.write("Match the re text in g(global):"+text.match(/re/g));
                        document.write("<br>Match the re text in i(case insensitive):"+text.match(/re/i));
                        document.write("<br>Match the re text in m(multile):"+text.match(/re/m));
                        document.write("<br>searching the regular in text:"+text.search(/regular/g));

                document.write("<h2>Regular expression match the range of characters</h2>");
                document.write("Match the any one chatcter range in [a-e]:"+text.match(/[a-e]/g));
                document.write("<br>Match the any one chatcter range not in [^a-e]:"+text.match(/[^a-e]/g));
                document.write("<br>Match the any one digits  range in [0-9]:"+text.match(/[0-9]/g));
                document.write("<br>Match the any one digits  range not in [0-9]:"+text.match(/[^0-9]/g));
                document.write("<br>Match the any one character in alternatein
(orange|pink):"+colors.match(/(pink|orange)/g));
                                        document.write("<h2>RegExp methods</h2>");
                        document.write("search string e using exe method:"+/o/.exec(text));
                document.write("<br>Check string e is existed or not usintest method:"+/o/.test(text));
                </script>
        </body>
</html>
```

Output:



RegExp Modifiers

Match the re text in g(global):re,re
Match the re text in i(case insensitive):re
Match the re text in m(multile):re
searching the regular in text:11

Regular expression match the range of characters

Match the any one chatcter range in [a-e]:e,c,e,e,a,e,e
Match the any one chatcter range not in [^a-e]:W,l,o,m, ,t,o, ,r,g,u,l,r, ,x,p,r,s,s,i,o,n, ,i,n, ,J,S
Match the any one digits range in [0-9]:null
Match the any one digits range not in [0-9]:W,e,l,c,o,m,e, ,t,o, ,r,e,g,u,l,a,r, ,e,x,p,r,e,s,s,i,o,n, ,i,n, ,J,S
Match the any one character in alternatein (orange|pink):pink,pink,pink,orange

RegExp methods

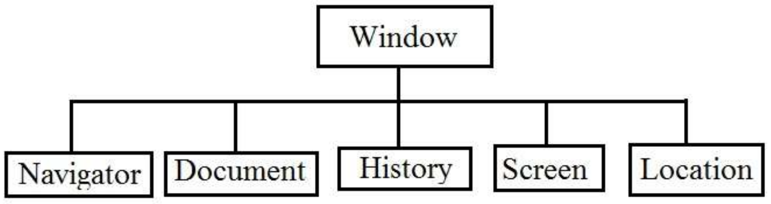
search string e using exe method:o
Check string e is existed or not usintest method:true

Working with Browser Objects

- The browser objects are automatically created by browser at the time of loading a Web Page.
- Browser objects are **Window, Navigator, Document, Screen, History,** and **Location**.

1. Window Object:

- The window object is used to open a window in a browser to display the web page.
- It is a global object, i.e that provides the global access to the associated variables and functions.
- Whenever you open a new window, a window object is created.
- Window object hierarchy of browser objects.



- The window objects has the following features
  - ✓ Window object collection
  - ✓ Window object properties
  - ✓ Window object methods

Window object collection:

- It is a set of all window objects available in HTML document.
- The window object contains collection of frames, which is used to retrieve a collection of window object defined in the specified document.

Window object properties:

- The properties used to retrieve the data about the window that is currently opened on the web browser.
- Syntax: **window . propertyname**

Property	**Description**
closed	Returns a Boolean true if a window is closed.
defaultStatus	Default message that has to be appeared in status bas. Deprecated.
document	Returns the Document object for the window.
frames	Returns all window objects running in the window.
History	Returns the History object for the window.
innerHeight	Returns the height of the window's content area (viewport) including scrollbars
innerWidth	Returns the width of a window's content area (viewport) including scrollbars
length	Returns the number of <iframe> elements in the current window
location	Returns the Location object for the window.
name	Sets or returns the name of a window
navigator	Returns the Navigator object for the window.
opener	Returns a reference to the window that created the window
outerHeight	Returns the height of the browser window, including toolbars/scrollbars
outerWidth	Returns the width of the browser window, including toolbars/scrollbars
parent	Returns the parent window of the current window
screenLeft	Returns the horizontal coordinate of the window relative to the screen
screenTop	Returns the vertical coordinate of the window relative to the screen
screenX	Returns the horizontal coordinate of the window relative to the screen

screenY	Returns the vertical coordinate of the window relative to the screen
self	Returns the current window
status	Specifies the message that is displayed in the status bar of a window. Deprecated.
top	Returns the topmost browser window

Window object methods:

- The methods are performing specific task to specified window.
- Example: window.alert("Alert message");

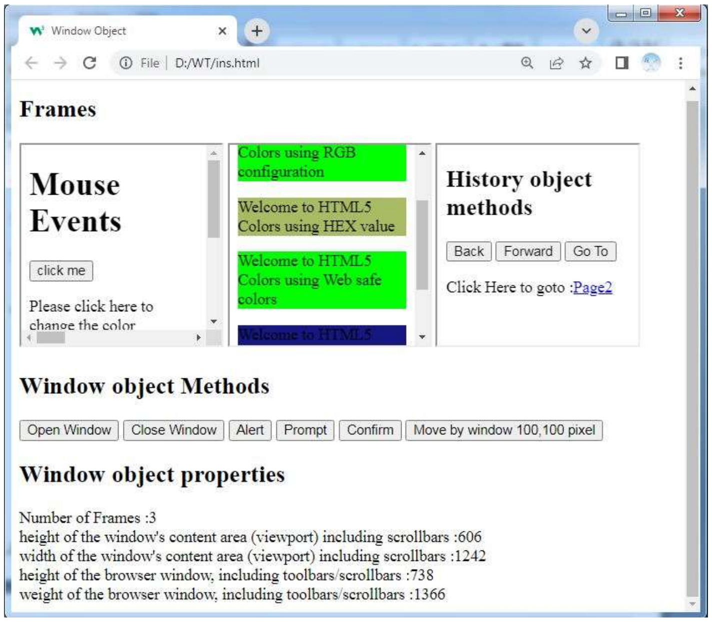
Method	**Description**
alert()	Displays an alert box with a message and an OK button
blur()	Removes focus from the current window
clearInterval()	Clears a timer set with setInterval()
clearTimeout()	Clears a timer set with setTimeout()
close()	Closes the current window
confirm()	Displays a dialog box with a message and an OK and a Cancel button
focus()	Sets focus to the current window
moveBy()	Moves a window relative to its current position
moveTo()	Moves a window to the specified position
open()	Opens a new browser window
print()	Prints the content of the current window
prompt()	Displays a dialog box that prompts the visitor for input
resizeBy()	Resizes the window by the specified pixels
resizeTo()	Resizes the window to the specified width and height
scroll()	Deprecated. This method has been replaced by the scrollTo() method.
scrollBy()	Scrolls the document by the specified number of pixels
scrollTo()	Scrolls the document to the specified coordinates
setInterval()	Calls a function or evaluates an expression at specified intervals (in millisec)
setTimeout()	Calls a function or evaluates an expression after a specified number of milli sec.
stop()	Stops the window from loading

Example: Write a JS to demonstrate the collections, properties and methods of Navigator objects

```
<!DOCTYPE HTML>
<html>
        <head>
                <title>Window Object</title>
        </head>
  <body>
                <h2>Frames</h2>
                <iframe src="bec.html" width="200" height="200"></iframe>
                <iframe src="colors.html" width="200" height="200"></iframe>
                <iframe src="" width="200" height="200"></iframe>
                <h2>Window object Methods</h2>
                <button id="but1" onClick="win = window.open()">Open Window</button>
                <button id="but1" onClick="win.close()">Close Window</button>
                <button id="but1" onClick="window.alert('Alert Message')">Alert</button>
                <button id="but1" onClick="window.prompt('prompt Message')">Prompt</button>
                <button id="but1" onClick="window.confirm('confirm Message')">Confirm</button>
        <button id="but1" onClick="win.moveBy(100,100)">Move by window 100,100 pixel</button>
                <h2>Window object properties</h2>
                <script type="text/javascript">
                        document.write("Number of Frames :"+window.length);
                        //chnage the location of frame 3
                        window.frames[2].location = "D://WT/page1.html";

                        document.write("<br>height of the window's content area (viewport) including
scrollbars :"+window.innerHeight);
                        document.write("<br>width of the window's content area (viewport) including
scrollbars :"+window.innerWidth);
                        document.write("<br>height of the browser window, including toolbars/scrollbars
:"+window.outerHeight);
                        document.write("<br>weight of the browser window, including toolbars/scrollbars
:"+window.outerWidth);
                </script>
        </body>
</html>
```

Output:



Frames

Mouse Events

click me

Please click here to change the color

Colors using RGB configuration

Welcome to HTML5 Colors using HEX value

Welcome to HTML5 Colors using Web safe colors

Welcome to HTML5

History object methods

Back Forward Go To

Click Here to goto :Page2

Window object Methods

Open Window | Close Window | Alert | Prompt | Confirm | Move by window 100,100 pixel

Window object properties

```
Number of Frames :3
height of the window's content area (viewport) including scrollbars :606
width of the window's content area (viewport) including scrollbars :1242
height of the browser window, including toolbars/scrollbars :738
weight of the browser window, including toolbars/scrollbars :1366
```

2. Navigator Object:

- Navigator object is used to display information about the version and type of the browser.
- The navigator objects has the following features
  - ✓ Navigator object collection
  - ✓ Navigator object properties
  - ✓ Navigator object methods

Navigator object collection:

- The navigator object is automatically created by the JavaScript run time system.
- The object contains all the information about the client browser.
- The navigator object provides the following collection objects
  - ✓ **plugins[]:** return a reference to all the embedded object in the document
  - ✓ **mimeTypes[]:** returns collection of MIME types supported by client browser.

Navigator object properties:

- Access the navigator properties as: **navigator . propertyname**

Property	Description
appCodeName	Returns browser code name
appName	Returns browser name
appVersion	Returns browser version
cookieEnabled	Returns true if browser cookies are enabled
platform	Returns browser platform
userAgent	Returns browser user-agent header

Navigator object methods:

- It refers to the function created inside the navigator object
- It is also used to check whether or not a browser supports java code.

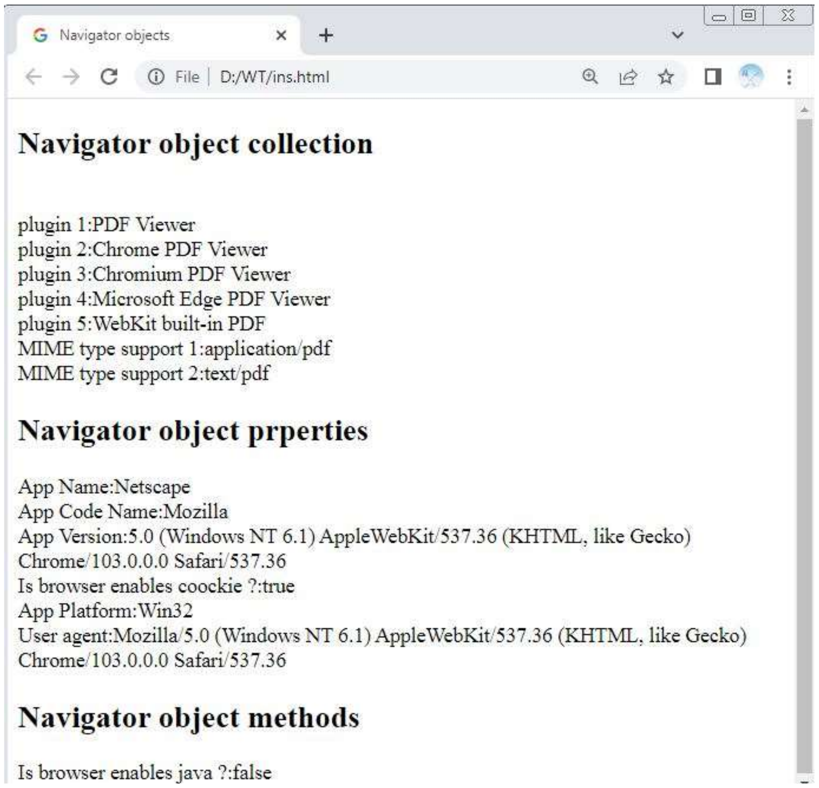
Method	Description
javaEnabled()	Returns true if the browser has Java enabled
taintEnabled()	Determines whether or not data tainting is enabled. Removed in JavaScript version 1.2 (1999).

Example: Write a JS to demonstrate the collections, properties and methods of Navigator object.

```
<!DOCTYPE html>
<html>
    <head>
            <title>Navigator objects</title>
    </head>
    <body >

            <script type="text/javascript">
                    var i;
                    document.write("<h2>Navigator object collection</h2>");
                    for(i=0;i<5;i++){
                            document.write("<br>plugin "+(i+1)+":"+navigator.plugins[i].name);
                    }
                    for(i=0;i<2;i++){
            document.write("<br>MIME type support "+(i+1)+":"+navigator.mimeTypes[i].type);
                    }
                    document.write("<h2>Navigator object prperties</h2>");
                    document.write("<br>App Name:"+navigator.appName);
                    document.write("<br>App Code Name:"+navigator.appCodeName);
                    document.write("<br>App Version:"+navigator.appVersion);
                    document.write("<br>Is browser enables coockie ?:"+navigator.cookieEnabled);
                    document.write("<br>App Platform:"+navigator.platform);
                    document.write("<br>User agent:"+navigator.userAgent);
                    document.write("<h2>Navigator object methods</h2>");
                    document.write("Is browser enables java ?:"+navigator.javaEnabled());
            </script>
    </body>
</html>
```

Output:



```
Navigator object collection

plugin 1:PDF Viewer
plugin 2:Chrome PDF Viewer
plugin 3:Chromium PDF Viewer
plugin 4:Microsoft Edge PDF Viewer
plugin 5:WebKit built-in PDF
MIME type support 1:application/pdf
MIME type support 2:text/pdf

Navigator object prperties

App Name:Netscape
App Code Name:Mozilla
App Version:5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/103.0.0.0 Safari/537.36
Is browser enables coockie ?:true
App Platform:Win32
User agent:Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/103.0.0.0 Safari/537.36

Navigator object methods

Is browser enables java ?:false
```

3. History Object:

- The history object contains an array of URL’s, which are visited by a user in the browser.
- The history objects has the following features
  - ✓ History object properties
  - ✓ History object methods

History object properties:

- Access the history properties as: **history . propertyname**

Property	Description
length	Returns the number of URLs (pages) in the history list
current	Specifies the URL of the current entry in the object.
next	Specifies the URL of the next element in the history list.
previous	Specifies the URL of the previous element in the history list.

History object methods:

- Methods are specifying the action related to the URL’s visited in the browser.

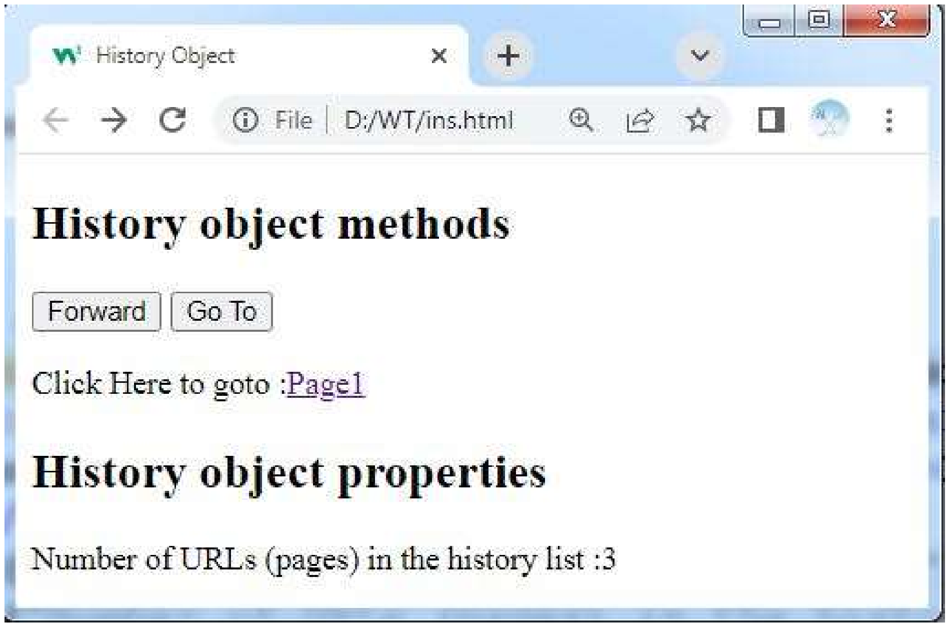
Method	Description
back()	Loads the previous URL (page) in the history list
forward()	Loads the next URL (page) in the history list
go()	Loads a specific URL (page) from the history list

Example: Write a JS to demonstrate the properties and method of History object.

```
<!DOCTYPE HTML>
<html>
        <head>
                <title>History Object</title>
        </head>
         <body>
                <h2>History object methods</h2>
                <button id="but2" onClick="window.history.forward()">Forward</button>
                <button id="but3" onClick="history.go(1)">Go To</button>
                <p>Click Here to goto :<a id="link" href ="page1.html">Page1</a></p>
                <script type="text/javascript">
                document.write("<h2>History object properties</h2>");
                document.write("Number of URLs (pages) in the history list :"+history.length);
                </script>
        </body>
</html>
```

Output:

Page1:



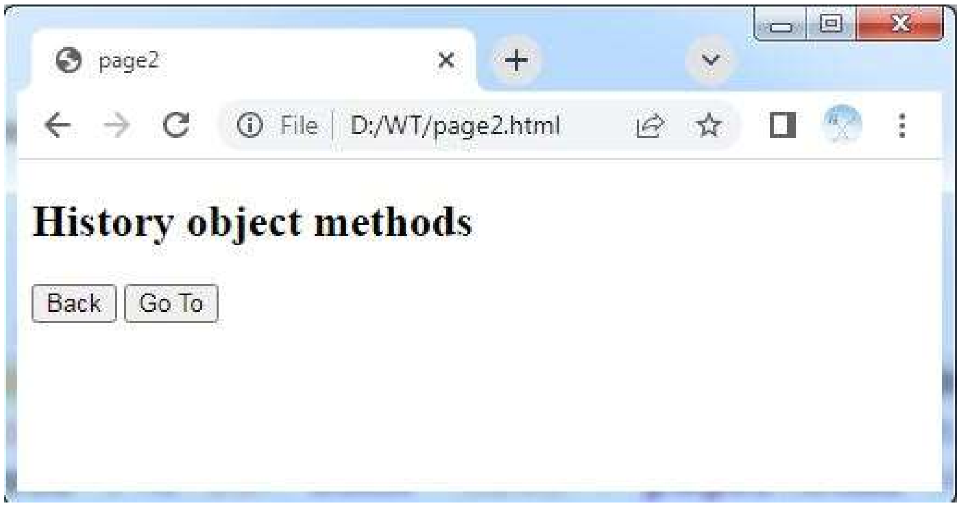
History object methods

Click Here to goto :Page1

History object properties

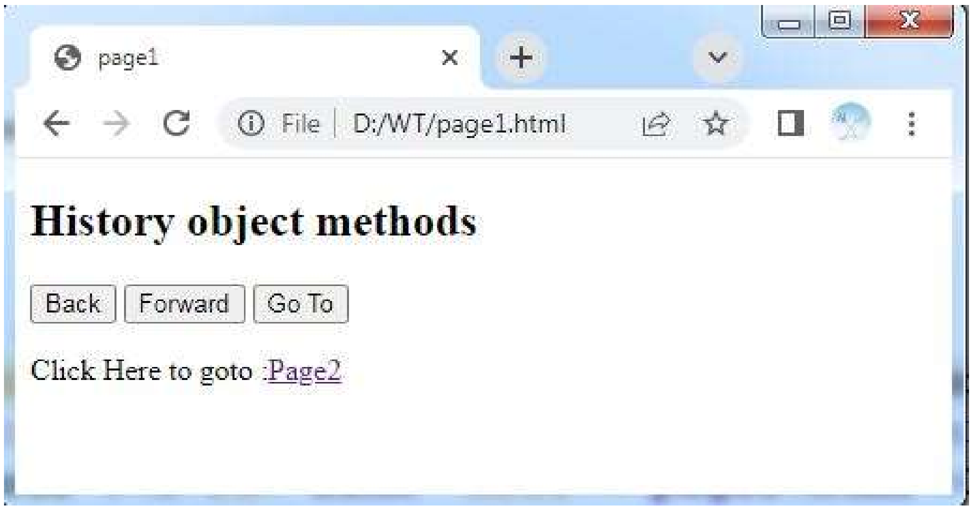
Number of URLs (pages) in the history list :3

Page2:



History object methods

Page3:



History object methods

Click Here to goto :Page2

4. Screen Object:

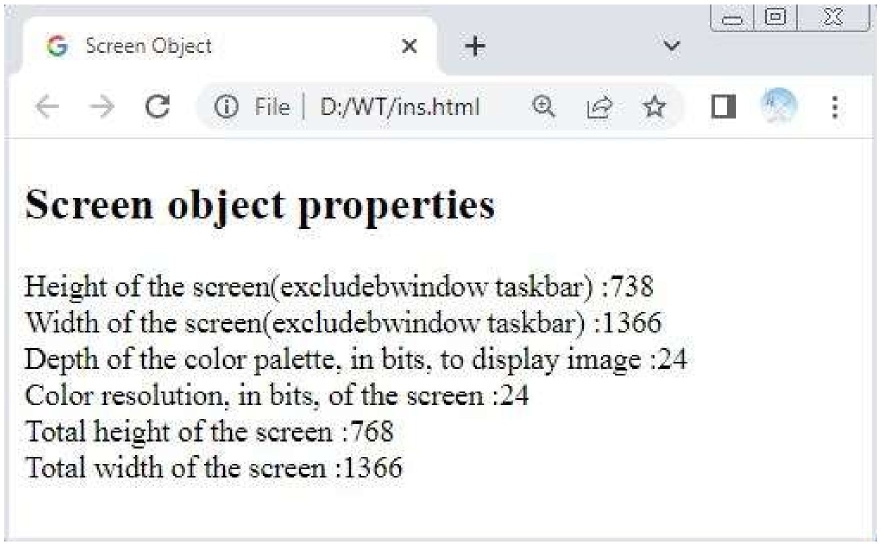
- The screen object provides the information about the dimensions of display screen, such as height, width, and color bit.
- Access the screen objcet as: **windows . screen (or) screen**
- Screen object having only properties
- Screen object properties are accessed by **screen . propertyname**

Property	**Description**
availHeight	Returns the height of the screen (excluding the Windows Taskbar)
availWidth	Returns the width of the screen (excluding the Windows Taskbar)
colorDepth	Returns the bit depth of the color palette for displaying images
Height	Returns the total height of the screen
pixelDepth	Returns the color resolution (in bits per pixel) of the screen
Width	Returns the total width of the screen

Example: Write a JS to demonstrate the properties Screen object.

```
<!DOCTYPE HTML>
<html>
    <head>
            <title>Screen Object</title>
    </head>
  <body>
            <script type="text/javascript">
            document.write("<h2>Screen object properties</h2>");
            document.write("Height of the screen(excludebwindow taskbar) :"+screen.availHeight);
            document.write("<br>Width of the screen(excludebwindow taskbar) :"+screen.availWidth);
    document.write("<br>Depth of the color palette, in bits, to display image :"+screen.colorDepth);
            document.write("<br>Color resolution, in bits, of the screen :"+screen.pixelDepth);
            document.write("<br>Total height of the screen :"+screen.height);
            document.write("<br>Total width of the screen :"+screen.width);
            </script>
    </body>
</html>
```

Output:



Screen object properties

```
Height of the screen(excludebwindow taskbar) :738
Width of the screen(excludebwindow taskbar) :1366
Depth of the color palette, in bits, to display image :24
Color resolution, in bits, of the screen :24
Total height of the screen :768
Total width of the screen :1366
```

5. Location Object:

- It is used to store the information of the current URL of the window object.
- The location object allows you to automatically navigate another web page.
- Example:

```
        <script>
                window.location = "https://google.com";
        </script>
```

- The location objects has the following features
  - ✓ Location object properties
  - ✓ Location object methods

Location object properties:

Property	Description
hash	Sets or returns the anchor part (#) of a URL
host	Sets or returns the hostname and port number of a URL
hostname	Sets or returns the hostname of a URL
href	Sets or returns the entire URL
pathname	Sets or returns the path name of a URL
port	Sets or returns the port number of a URL
protocol	Sets or returns the protocol of a URL
search	Sets or returns the querystring part of a URL

Location object methods:

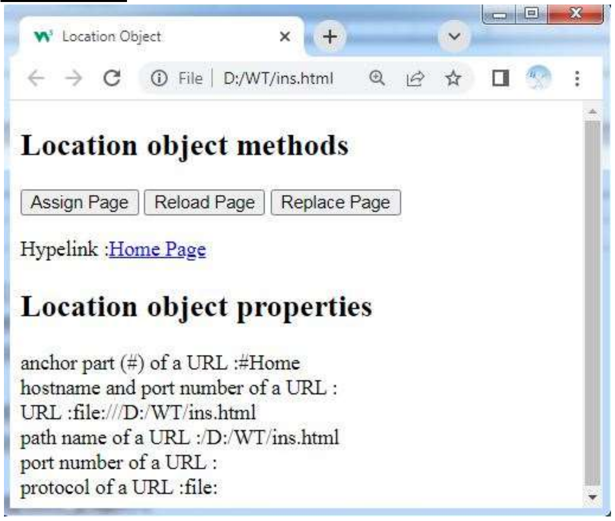
Method	Description
assign()	Loads a new document in the browser
reload()	Reloads the current document that is contained in location.href property
replace()	Replaces the current document with a new one. You con't move back to previous document using the browser back button.

Example: Write a JS to demonstrate the properties and methods of Location object.

```
<!DOCTYPE HTML>
<html>
    <head>
            <title>Location Object</title>
    </head>
    <body>
            <h2>Location object methods</h2>
            <button id="but1" onClick="assign_page()">Assign Page</button>
            <button id="but2" onClick="reload_page()">Reload Page</button>
            <button id="but3" onClick="replace_page()">Replace Page</button>
            <p>Hypelink :<a id="link" href ="index.htm#Home">Home Page</a></p>
            <script type="text/javascript">
            document.write("<h2>Location object properties</h2>");
            var loc = document.getElementById("link");
            document.write("anchor part (#) of a URL :"+loc.hash);
            document.write("<br>hostname and port number of a URL :"+location.host);
            document.write("<br>URL :"+location.href);
            document.write("<br>path name of a URL :"+location.pathname);
```

```
                    document.write("<br>port number of a URL :"+location.port);
                    document.write("<br>protocol of a URL :"+location.protocol);
                    function assign_page(){
                            location.assign("colors.html");
                    }
                    function reload_page(){
                            location.reload();
                    }
                    function replace_page(){
                            location.replace("bec.html");
                    }
                    </script>
            </body>
</html>
```

Output 1:



Location object methods

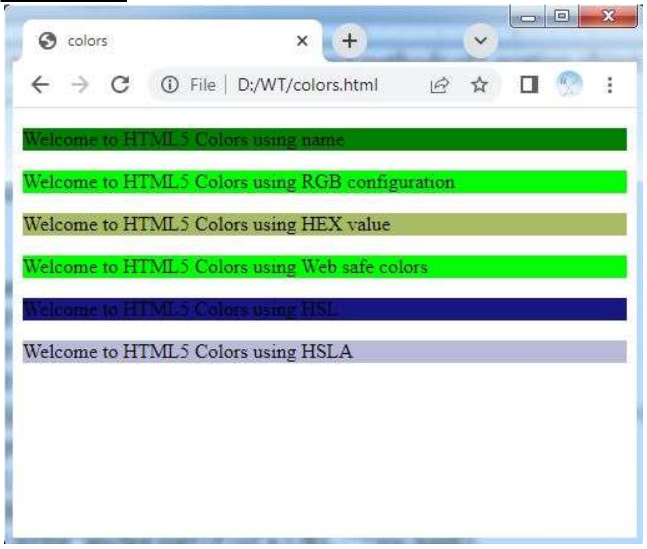
Assign Page | Reload Page | Replace Page

Hypelink :Home Page

Location object properties

```
anchor part (#) of a URL :#Home
hostname and port number of a URL :
URL :file:///D:/WT/ins.html
path name of a URL :/D:/WT/ins.html
port number of a URL :
protocol of a URL :file:
```

Output 2:



Welcome to HTML5 Colors using name

Welcome to HTML5 Colors using RGB configuration

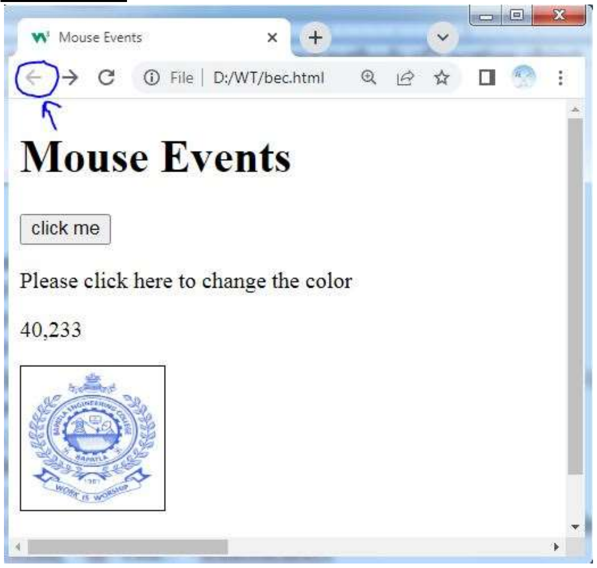
Welcome to HTML5 Colors using HEX value

Welcome to HTML5 Colors using Web safe colors

Welcome to HTML5 Colors using HSL

Welcome to HTML5 Colors using HSLA

Output 3:



Mouse Events

click me

Please click here to change the color

40,233

Working with Document Object

- The document object is child object of window object, which refers to a browser.
- The document object is created when HTML document is loaded in a browser.
- The document object is an object that provides to access the **all the HTML elements of a document**.
- The document object stores the elements of HTML document, such as HEAD, BODY, and HTML as objects.
- You can access document object either w**indow.document** or **document** properties.
- The document object has the following features
  - Document object collection
  - Document object properties
  - Document object methods
- **Document object collection:**
- Collection is defined as an array of related objects.

Collection	Description	Synatx
Forms	Returns all <form> elements	doctype.forms[]
Images	Returns all <img> elements	doctype.images[]
Links	Returns all <area> and <a> elements that have a href attribute	doctype.links[]

- **Document object properties:**
- The property is accessed by **objectName . propertyName**

Property	Description
Cookie	Returns the document's cookie
Domain	Returns the domain name of the document server
lastModified	Returns the date and time the document was updated
documentMode	Return the mode used by the browser to process the document
readyState	Returns the (loading) status of the document
Referrer	Returns the URI of the referrer (the linking document)
Title	Returns the <title> element
URL	Returns the complete URL of the document

- **Document object methods:**
- The method is call by **objectName . methodName(arguments)**

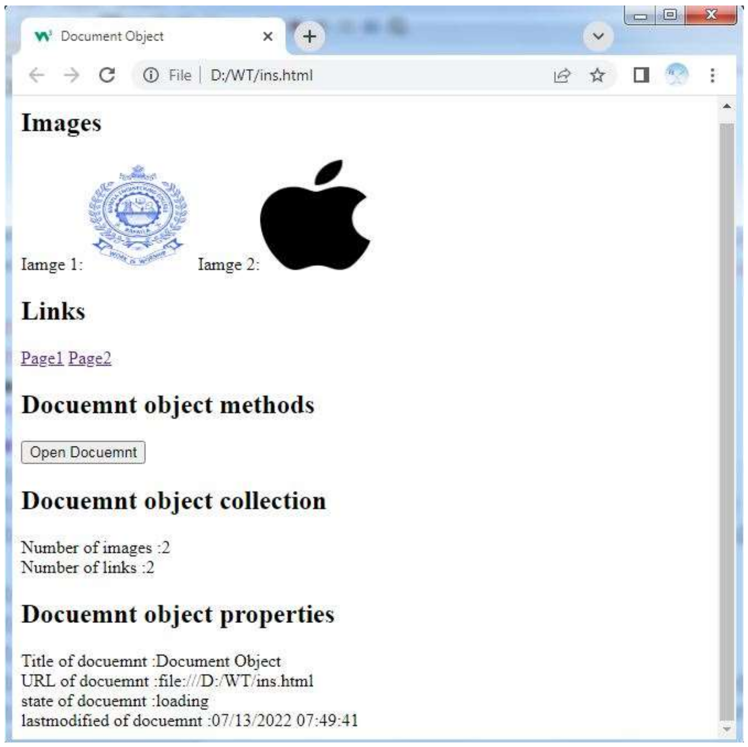
Method	Description
open()	Opens an HTML document to display the output
close()	Closes the HTML document
getElementById()	Returns the element that has the ID attribute with the specified value
getElementsByName()	Returns an live NodeList containing all elements with the specified name
getElementsByTagName()	Returns an HTMLCollection containing all elements with the specified tag name
getElementsByClassName()	Returns an HTMLCollection containing all elements with the specified class name
write()	Writes HTML expressions or JavaScript code to a document
writeln()	Same as write(), but adds a newline character after each statement

Example: Write a JS to demonstrate the properties and methods of document object.

```
<!DOCTYPE HTML>
<html>
    <head>
            <title>Document Object</title>
    </head>
  <body>
            <h2>Images</h2>
            Iamge 1:<img id ="img1" src="images/beclogo.png" width="100" height="100"/>
            Iamge 2:<img id ="img2" src="images/applogo.png" width="100" height="100"/>
            <h2>Links</h2>
            <a id ="link1" href="bec.html" >Page1</a>
            <a id ="link2" href="page2.html">Page2</a>
            <h2>Docuemnt object methods</h2>
            <button onclick="open_doc()">Open Docuemnt</button>
            <script type="text/javascript">
                    document.write("<h2>Docuemnt object collection</h2>");
                    document.write("Number of images :"+document.images.length);
                    document.write("<br>Number of links :"+document.links.length);

                    document.write("<h2>Docuemnt object properties</h2>");
                    document.write("Title of docuemnt :"+document.title);
                    document.write("<br>URL of docuemnt :"+document.URL);
                    document.write("<br>state of docuemnt :"+document.readyState);
                    document.write("<br>lastmodified of docuemnt :"+document.lastModified);
                    function open_doc(){
                            var mywindow = window.open();
                            mywindow.document.open();
                            mywindow.document.write("<h1>Welcome to document</h1>");
                            mywindow.document.close();
                    }
            </script>
    </body>
</html>
```

Output:



Images

Iamge 1: Iamge 2:

Links

Page1 Page2

Docuemnt object methods

Open Docuemnt

Docuemnt object collection

Number of images :2
Number of links :2

Docuemnt object properties

Title of docuemnt :Document Object
URL of docuemnt :file:///D:/WT/ins.html
state of docuemnt :loading
lastmodified of docuemnt :07/13/2022 07:49:41

Document Object Model (DOM)

- The **D**ocument **O**bject **M**odel(DOM) is a cross-platform and language-independent interface that allows programs and script to dynamically access and update the content, structure, and style of HTML or XML documents.
- You will learn all about JavaScript DOM in the following:
  1. JavaScript DOM Nodes
  2. JavaScript DOM Levels
  3. JavaScript DOM Interfaces

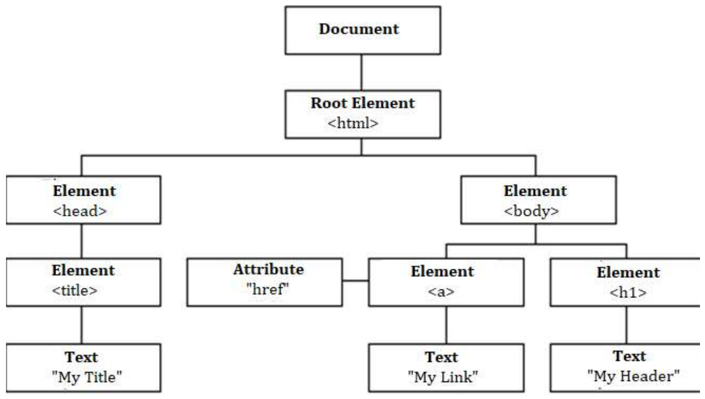
1. Understanding DOM Nodes:

- Every element in an HTML page represents a DOM node.
- These elements are related to each other through the parent-child relationship.
- An element inside another element is known as child element or child node.
- An element that contains the element within it is knows the parent element or parent node.
- Example1:<p> Welcome to DOM</p>

  Here P is parent and having one child node and Text (Welcome to DOM) is child node.
- Example2:<p> Welcome to <B>DOM</B></p>
- Here P is parent and having two child nodes (B and Text (Welcome to))

  B node also contains child node Text (DOM).
- **Example3:**

```
<!DOCTYPE html>
<html>
        <head>
                <title>My Title</title>
        </head>
        <body>
                <h1>My Header</h1>
        <a href="bec.html">My Link</a>
        </body>
</html>
```

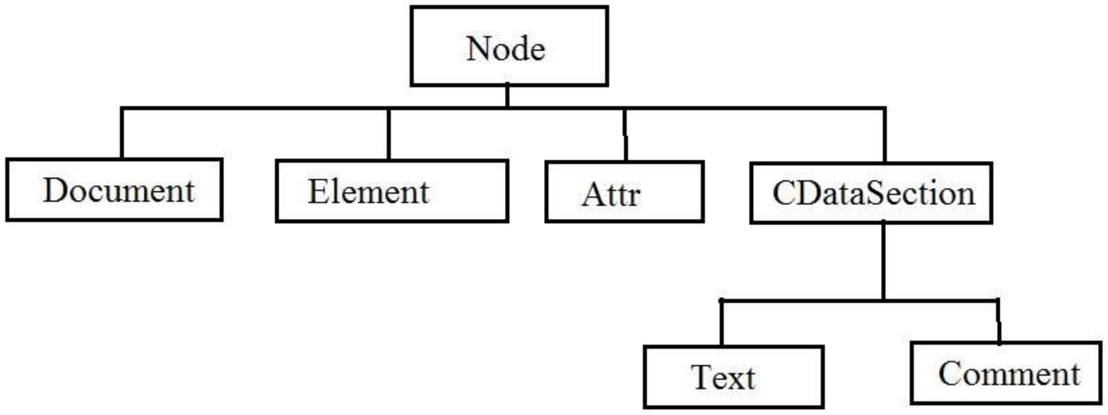


Displaying the Node Tree of an HTML Document

- The key components of the node structure are
  - ✓ **Element nodes:** Represents the basic building blocks of document know as elements. These elements contains other elements, such as HTML, HEAD, BODY, A, AND H1.
  - ✓ **Attribute nodes:** Provides more information about elements. Attribute elements are always inside the element nodes. Example 'href' attribute inside 'A' element.
  - ✓ **Text nodes:** Represents the content contained in element nodes, such as "My Title", "My Link", "My Header"
- Every node has some properties that contain information about the node.
- The properties are
  - ✓ **nodeName:** contains the name of the node. The **nodeName** properties for the following nodes are

    Element node: Represent the element name
    Attribute node: Represent the attribute name
    Text node: Represent the Text
  - ✓ **nodeValue:** Represents the value of the node.
  - ✓ **nodeType:** Represents the type of the node.

- Displaying the Node Tree of an HTML Document according to inheritance hierarchy



- DOM Nodes and their numeric values are as follow

Node Type	**Numeric value**
Element	1
Attr	2
Text	3
CDataSection	4
Comment	8
Document	9

Example: Write a JS program to check the type of node in DOM.

```
<!DOCTYPE HTML>
        <HEAD>
                <TITLE>checking the node type</TITLE>
                <SCRIPT type="text/javascript">
                        function checkNodeType()    {
                                var tag=document.getElementById("T1");
                                var value= tag.nodeType;
                                if(value== 1)
                                        document.write("This is the Element Node type");
                                if(value== 2)
                                        document.write("This is the Attribute Node type");
                                if(value== 3)
                                        document.write("This is the Text Node type");
                                if(value== 8)
                                        document.write("This is the Comment Node type");
                                if(value== 9)
                                        document.write("This is the Document Node type");
                        }
                </SCRIPT>
        </HEAD>
        <BODY>
                <H1>Checking the DOM Node Type</H1>
                <P id="T1"> Welcome to the world of DOM</P>
                <INPUT type= "button" onclick="checkNodeType();"  value="Check Node Type!">
        </BODY>
</HTML>
```

2. Understanding DOM Level:

- The W3C DOM specifications are divided into different levels where each level contains some required and optional modules.
- **Levels of DOM:** Here the following table describes levels of DOM.

Level	Description
Level 0	supports an intermediate DOM, which exists before the creation of DOM Level 1
Level 1	It is an API that allows program and scripts to dynamically access and update the content, structure, and style of HTML and XML 1.0 documents
Level 2	extends Level 1 with support for XML 1.0 with namespaces and adds support for CSS, events, and enhances tree manipulations
Level 3	extends Level 2 by finishing support for XML 1.0 with namespaces

- **JavaScript DOM Level 2:** The JavaScript DOM Level 2 defines the following specifications, which have reached in their final form

Specification	Description
DOM Level 2 Core	helps the programs and scripts to access and update the content and structure of document dynamically
DOM Level 2 Views	allows programs and scripts to dynamically access and update the content of an HTML or XML document
DOM Level 2 Events	provides a generic event system to programs and scripts
DOM Level 2 Style	allows programs and scripts to dynamically access and update the content of style sheets
DOM Level 2 Traversal and Range	allows programs and scripts to dynamically traverse and identify a range of content in a document
DOM Level 2 HTML	allows programs and scripts to dynamically access and update the content and structure of HTML 4.01 and XHTML 1.0 documents

- **JavaScript DOM Level 3:** Basically DOM defined six specifications at Level 3, which are:
  - DOM Level 3 Core
  - DOM Level 3 Load and Save
  - DOM Level 3 XPath
  - DOM Level 3 Views and Formatting
  - DOM Level 3 Requirements
  - DOM Level 3 Validation
- But only three of them were recommended by W3C for public use, all the three are described in the following table.

Specification	Description
DOM Level 3 Core	allows programs and scripts to dynamically access and update the content, structure, and document style
DOM Level 3 Load and Save	allows programs and scripts to dynamically load the content of an XML document into a DOM document
DOM Level 3 Validation	allows programs and scripts to dynamically update the content and structure of documents and ensures that the document remains valid

3. Understanding DOM Interfaces:

- The DOM API provides interfaces to implement DOM. Each interface is associated with a particular type of node that is defined in the inheritance hierarchy, which are described in the following table:
  - The Node Interface
  - The Document Interface
  - The Attr Interface
  - The Element Interface

The Node Interface:

- This interface is the base interface in the DOM tree. It means that all the other interfaces are derived from the Node interface.
- The Node Interface has **Properties** and **Methods**
- The table given below describes properties of Node interface in JavaScript.

Property	Description
nodeType	returns an integer value that represents the node type
nodeName	returns the name of the node
nodeValue	returns the value of the node
attributes	returns an array of Attr objects that represents the attributes of the current node
parentNode	returns the parent node of the current node if it exists, otherwise it returns a null value
childNodes	returns an array of Node objects that represents the child nodes of the current node
firstChild	returns a Node object that represents the first child of the current node, otherwise it returns a null value
lastChild	returns a Node object that represents the last child of the current node, otherwise it returns a null value
localName	returns a qualified name of the current node
namespaceURI	returns the namespace URI of the current node
nextSibling	returns the node that immediately follows the current node
ownerDocument	returns a Document object associated with the current node
prefix	returns the namespace prefix of the current node
prviousSibling	returns the node immediately preceding the current node

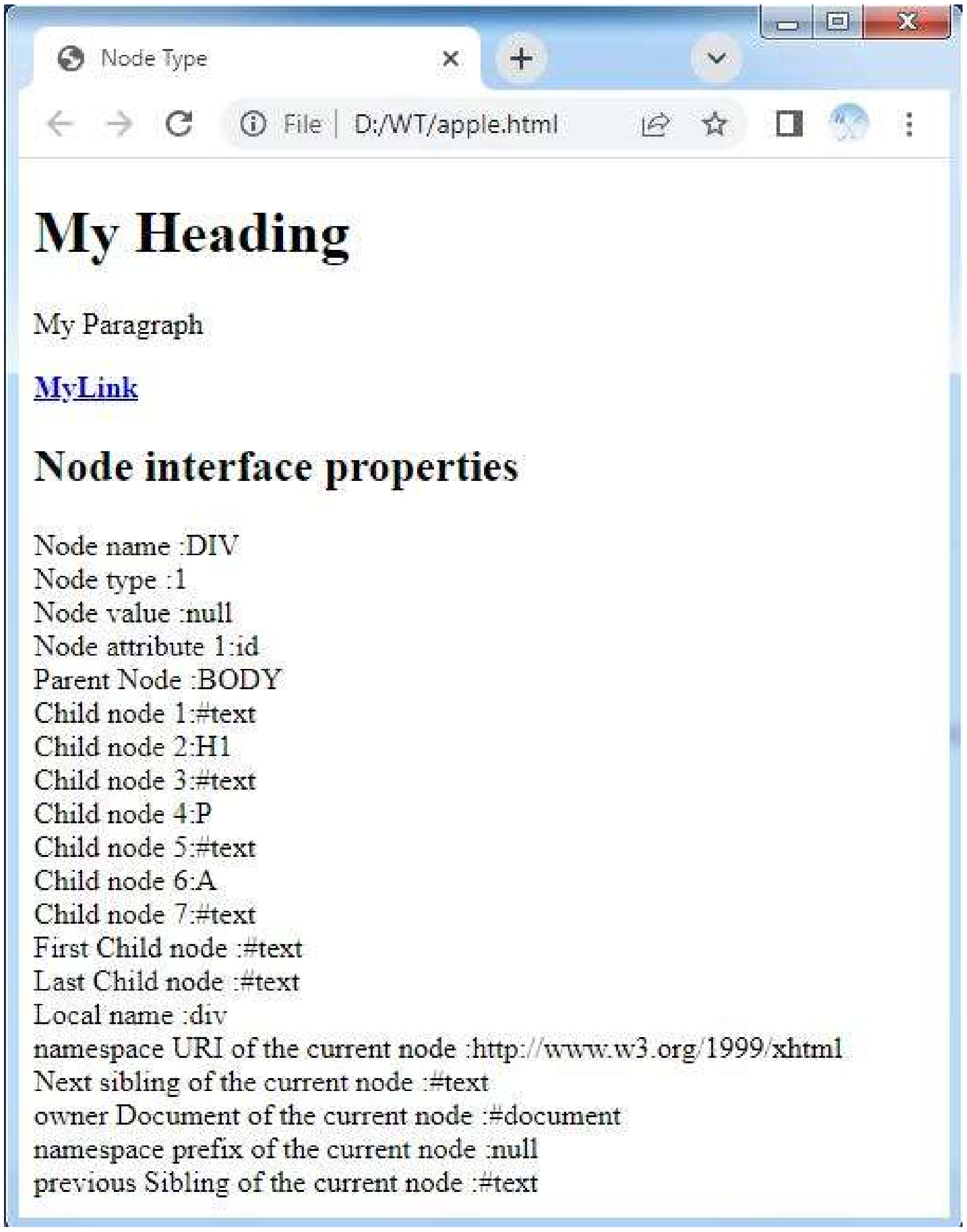
- The following table lists the methods of the Node interface in JavaScript.

Method	Description
createElement()	creates a new element of the node
appendChild()	adds a new child node at the end of the list of the child nodes
cloneNode()	creates a duplicate node of the current node
hasAttributes()	returns true if the node has an attribute
hasChildNodes()	returns true if the node has child nodes
insetBefore()	inserts a new child node before the current child node
inSupported()	checks whether a specified feature is implemented by the DOM implementation and is supported by the current node
normalize()	places the sub-tree of the current node in state where the structural nodes separated by the text nodes
removeChild()	removes an existing child node from the list of child nodes
replaceChild()	replaces an existing child node with the new node

Example: Write a JS to demonstrate the properties of Node interface.

```
<!DOCTYPE HTML>
<html>
        <head>
                <title>Node Type</title>
        </head>
        <body >
                <div id="d">
                        <h1>My Heading</h3>
                        <p id="para">My Paragraph</p>
                        <a id="link" href="bec.html"><b>MyLink</b></a>
                </div>
                <script type="text/javascript">
                        var anchor = document.getElementById("d");
                        document.write("<h2>Node interface properties</h2>");
                        document.write("Node name :"+anchor.nodeName);
                        document.write("<br>Node type :"+anchor.nodeType);
                        document.write("<br>Node value :"+anchor.nodeValue);
                        for(var i=0;i<anchor.attributes.length; i++) {
                                document.write("<br>Node attribute
                                                "+(i+1)+":"+anchor.attributes[i].nodeName);
                        }
                        document.write("<br>Parent Node :"+anchor.parentNode.nodeName);
                        for(var i=0;i<anchor.childNodes.length; i++) {
                                document.write("<br>Child node
                                               "+(i+1)+":"+anchor.childNodes[i].nodeName);
                        }
                        document.write("<br>First Child node :"+anchor.firstChild.nodeName);
                        document.write("<br>Last Child node :"+anchor.lastChild.nodeName);
                        document.write("<br>Local name :"+anchor.localName);
                        document.write("<br>namespace URI of the current node
                                      :"+anchor.namespaceURI);
                        document.write("<br>Next sibling of the current node
                                       :"+anchor.nextSibling.nodeName);
                        document.write("<br>owner Document of the current node
                                  :"+anchor.ownerDocument.nodeName);
                         document.write("<br>namespace prefix of the current node
                         :"+anchor.prefix);
                        document.write("<br>previous Sibling of the current node
                                  :"+anchor.previousSibling.nodeName);
                </script>
        </body>
</html>
```

Output:



My Heading

My Paragraph

MyLink

Node interface properties

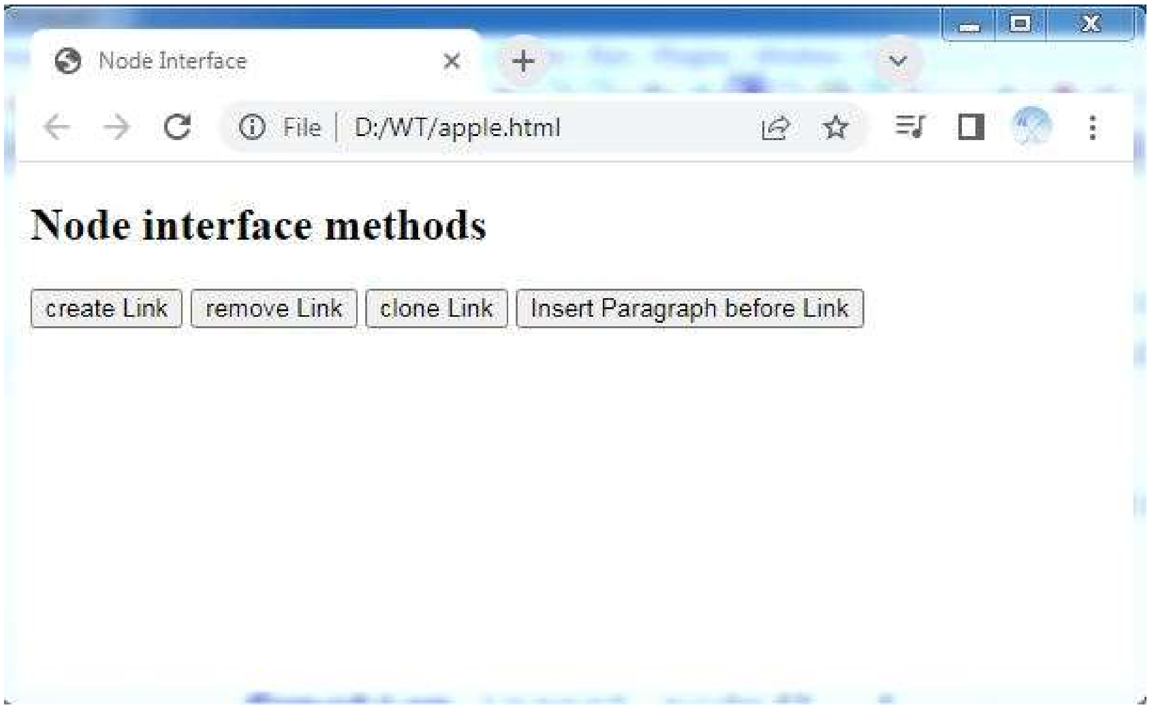
```
Node name :DIV
Node type :1
Node value :null
Node attribute 1:id
Parent Node :BODY
Child node 1:#text
Child node 2:H1
Child node 3:#text
Child node 4:P
Child node 5:#text
Child node 6:A
Child node 7:#text
First Child node :#text
Last Child node :#text
Local name :div
namespace URI of the current node :http://www.w3.org/1999/xhtml
Next sibling of the current node :#text
owner Document of the current node :#document
namespace prefix of the current node :null
previous Sibling of the current node :#text
```

Example: Write a JS to demonstrate the methods of Node interface.

```
<!DOCTYPE HTML>
<html>
	<head>
		<title>Node Interface</title>
		<script type="text/javascript">
			function create_node()  {
				var link = document.createElement("a");
				link.setAttribute("id","l1");
				link.setAttribute("href","D://images/beclogo.png");
				link.setAttribute("width","100");
				link.setAttribute("height","100");
				text = document.createTextNode("My Link");
				link.appendChild(text);
				document.body.appendChild(link);
			}
			function delete_node()  {
				child = document.getElementById("l1");
				document.body.removeChild(child);
			}
			function clone_node()  {
				link = document.getElementById("l1");
				c = link.cloneNode(true);
				document.body.appendChild(c);
			}
			function insert_node()  {
				link = document.getElementById("l1");
				para = document.createElement("p");
				para.setAttribute("id","p1");
				text = document.createTextNode("My Paragraph before link");
				para.appendChild(text);
				document.body.insertBefore(para,link);
			}
		</script>
	</head>
	<body >
		<div id="d">
			<h2>Node interface methods</h2>
			<button onclick="create_node()">create Link</button>
			<button onclick="delete_node()">remove Link</button>
			<button onclick="clone_node()">clone Link</button>
			<button onclick="insert_node()">Insert Paragraph before Link</button>
		</div>
	</body>
</html>
```

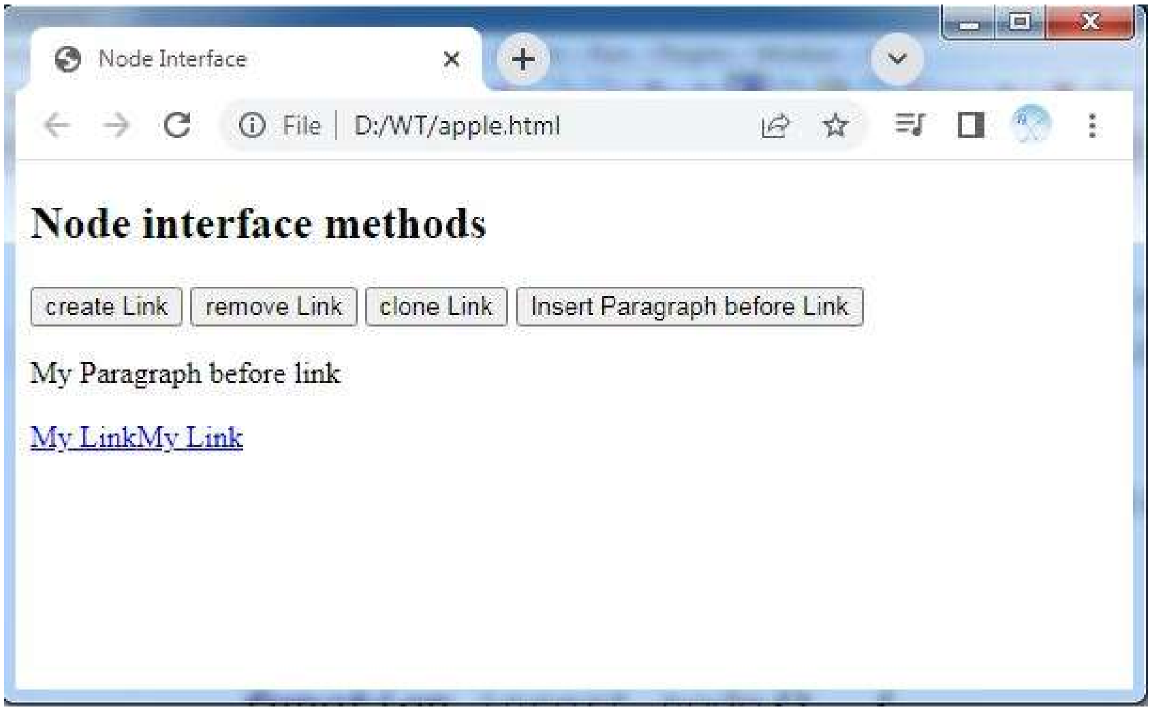
Output:

Before creating, cloning and insert paragraph



Node interface methods

After creating, cloning and insert paragraph



Node interface methods

My Paragraph before link

My LinkMy Link

The Document Interface:

- This interface is the root node in the DOM tree. It defines the documentElement property of the document, which returns the root node by looping through all the child nodes of the Document nodes to test whether or not they are Element nodes
- The following table describes methods of Document interface in JavaScript.

Method	**Description**
getElementByTagName()	returns an array of element objects that have a tag name similar to the specified name
getElementByTagNameNS	returns an array of an element object whose name attribute's value matches the value of the argument of this method
getElementById()	returns the element object that has the same id, as specified in the argument

The Element Interface:

- This interface represents an element in an HTML document.
- Here is the table describes methods of the Element interface in JavaScript.

Method	**Description**
hasAttribute()	returns true if the specified element has an attribute
getAttribute()	returns the value of the attribute of the specified element
setAttribute()	assigns the specified value to the specified attribute
removeAttribute()	removes the specified attribute
getElementByTagName()	returns an array of element objects whose name matches with that provided in the tagName attribute

The Attr Interface:

- This interface represents an attribute of an element. It inherits the Node interface.
- The following table describes properties of the Attr interface in JavaScript.

Property	**Description**
name	specifies the name of the attribute
value	specifies the value of the attribute
ownerElement	returns the element node of the Attr object
specified	returns true if the attribute is set by the user or returns false if the attribute is set by the default value

One mark Questions:

1. Write the syntax for function template object creation.
2. List the browser objects.
3. List the java script objects.
4. Write the methods of number object.
5. Write the methods of Array object.
6. Write the properties of Navigator object
7. What is document object collection?
8. List the methods to get the elements from document.
9. What is DOM.
10. Write the properties of node.
11. Write the syntax to create an element in DOM.
12. Write the syntax to create a text node in DOM.

Essay Questions:

1. Explain String object in JavaScript with an example.
2. Explain Date object in JavaScript with an example.
3. Explain Window object in JavaScript with an example.
4. Explain Document object in JavaScript with an example.
5. Explain DOM node tree structure with an example.
6. Explain DOM node interface in JavaScript with an example.

Programming Exercises:

1. Write a JavaScript program that to find the biggest among three numbers.
2. Write a JavaScript program to check whether the given number is prime or not.
3. Write a JavaScript program that demonstrates popup windows.
4. Write a JavaScript program that demonstrates the timer functions.
5. Write a JavaScript program to perform the following operation using DOM.
   i) Create element ii) Remove element iii) Clone element iv) insert before