Java Script

- **Web development**: Java Script is client side scripting language (programming) used for web development. i.e., dynamic and interactive elements to websites.
- **Lightweight interpreted programming language**
- **Java script is a object-based programming language**
- It supports cross-platform
- Developed in Net Scape
- What is client side scripting language? :
  - ✓ The *source code is processed by the client's web browser rather than on the web server*.
  - ✓ The *java script functions run after a webpage has loaded without communicating with the server*.
  - ✓ JavaScript code can *produce an error message before any information is actually transmitted to the server.*
- JavaScript code can be *inserted anywhere within the HTML of a webpage*.

Applications of Java Script:

1. **Client side validation** - This is really important to verify any user input before submitting it to the server and Javascript plays an important role in validting those inputs at front-end itself.
2. **Manipulating HTML Pages** - Javascript helps in manipulating HTML page on the fly. This helps in adding and deleting any HTML tag very easily using javascript and modify your HTML to change its look and feel based on different devices and requirements.
3. **User Notifications** - You can use Javascript to raise dynamic pop-ups on the webpages to give different types of notifications to your website visitors.
4. **Back-end Data Loading** - Javascript provides Ajax library which helps in loading back-end data while you are doing some other processing. This really gives an amazing experience to your website visitors.
5. **Presentations** - JavaScript also provides the facility of creating presentations which gives website look and feel. JavaScript provides RevealJS and BespokeJS libraries to build a web-based slide presentations.

6. **Server Applications** - Node JS is built on Chrome's Javascript runtime for building fast and scalable network applications. This is an event based library which helps in developing very sophisticated server applications including Web Servers.

- **Syntax for Define Java Script**:

```
<script type=”type/javascript”>
  // content of java script code
</script>
```

- **Note**: <scrip> tag can be placed either in <head> or <body> section in HTML document

Java script code to change the font size: (id name of the <p> is “demo”)

```
document.getElementById('demo').style.fontSize='35px'"
```

Logic to change the src attribute of image tag:

```
document.getElementById('myImage').src='pic_bulbon.gif'
```

Hiding of Text:

```
document.getElementById("demo").style.display = "none";
```

Display of the text:

```
document.getElementById("demo").style.display = "block";
```

External Scripting Language:

1. Save the javascript source file with .js extension
2. Include the .js file with <script src=”filename.js”> </script>

Displaying the content:

1. Writing into an HTML element, using innerHTML.
2. Writing into the HTML output using document.write().
3. Writing into an alert box, using window.alert().
4. Writing into the browser console, using console.log().

Document Object Model (DOM)

- Object Hierarchy: The documents are represented in hierarchy manner
- **Purpose of DOM**:
  1. **Programming Interface:** It provides the programming interface to HTML and XML.
  2. **Logical structure:** It defines the logic structure of a document
  3. **Access and Manipulation of documents**: It defines the way the document is accessed and manipulated (create, modify and remove elements in the page dynamically).
- **DOM nodes**: Object which is returned by getElementById(:Id name")
- **DOM tree**: All the nodes in a document represented in a tree format (child, parent, siblings properties)
- **DOM object**:
  1. **Window** – it represents the browser window and access the document object contained in the window. Also contains history and location objects.
  2. **Document** – XHTML document rendered in a window. Accessing the every element in a document and allows dynamic modifications.
  3. **Body** – access the body elements of an XHTML document
  4. **History** – Keep the track of the sites visited by the browser.
  5. **Location** – contains the URL of the rendered document. (set to new URL, the browser immediately navigates to the new window)
- **DOM Collections**: The document object has the following properties
  1. **Image collections** (<img>)
     Var x=document.forms (property : length)
  2. **Form collections** (<form>)
     var x=document.forms; (all attributes of form tag)
  3. **Anchor collections** (<a>)
     var x=document.anchors;
     attributes: length, innerHTML, name, href
     accessing values: document.anchors[i].attribute;

Methods:

i. [index] – returns the <a> element from the collection with the specified index. (start from index 0. Returns null if the index is out of range)

ii. Item(index) - returns the <a> element from the collection with the specified index. (start from index 0. Returns null if the index is out of range)

iii. namedItem(id) – returns the <a> element with the specified id (returns null if the id is not exist)

4. Link Elements (<a href>)

   var x=document.links;

   attributes: length, innerHTML, name, href

❖ ***Methods* of Document Object**:

1. **write("string"):** writes the given string on the document.
2. **getElementById():** returns the element having the given id value.
3. **getElementsByName():** returns all the elements having the given name value.
4. **getElementsByTagName():** returns all the elements having the given tag name.
5. **getElementsByClassName():** returns all the elements having the given class name.

Others DOM Comments:

- ❖ find the cookies: document.cookie
- ❖ Find the domain address: document.domain
- ❖ Find the referrer of the document (find the URl of the current document) : document.referrer
- ❖ Find the current URL of the document : document.URL
- ❖ **Logic for open a new window and define the content**:

```
var w = window.open();
w.document.open();
w.document.write("<h1>Hello World!</h1>");
w.document.close();
```

Example Program:

Logic for hiding the elements by using interactions:

```
<html>
```

```
<head>
</head>
<body >
<a href="dom.html"> CSE </a>
<p class="a"> welcome</p>
<p onclick="s();"> to</p>
<a href="dom.html"> Department  </a>
<p id = "a"> CSE </p>
<a href="dom.html"> Welcome </a>
<script type="text/javascript">
function s()
{
document.getElementById("a").style.visibility="hidden";
}
</script>
</body>
</html>
```

Display all the tag information:

```
<html>
<head>
</head>
<body>
<p> welcome</p>
<p> to</p>
<p> CSE </p>
<script type="text/javascript">
var x, i, l;
x = document.getElementsByTagName("*");
l = x.length;
for (i = 0; i < l; i++) {
  document.write(x[i].tagName + "<br>");
}
</script>
```

```
</body>
</html>
```

getElementByClassName:

```
<html>  <head>  </head>
<body>
<p class="a"> welcome</p>
<p> to</p>
<p class = "a"> CSE </p>
<script type="text/javascript">
var x, i, l;
x = document.getElementsByClassName("a");
l = x.length;
for (i = 0; i < l; i++) {
  document.write(x[i].tagName + "<br>");
  x[i].style.color="red";
} </script>  </body>  </html>
```

Creation of New Elements By using DOM

- **Creation of new elements**: (appended at the end in a body)

```
<html>
<head>
<script type="text/javascript">
function s()
{
   var d=document.createElement("Button");
   d.innerHTML = "Text Me";
   document.body.appendChild(d);
}
</script>  </head>  <body>
<h1 onclick="s();">Welcome </h1>
</body>  </html>
```

- **Creation of new nodes in anywhere of the document**:

```
<html>
```

```
<head>
<script type="text/javascript">
function s()
{
   var d=document.createElement("input");
   d.type = "Text";
   document.getElementById("hh").appendChild(d);
}
</script>
</head>
<body>
<h1 onclick="s();">Welcome </h1>
<p id="hh"> CSE department </p>
<big> Governemtn college of Engineering </big>
</body>
</html>
```

Create text Node for creating text for tags:

```
<html>
<head>
<script type="text/javascript">
function s()
{
   var d=document.createElement("p");
   var t =document.createTextNode("I am doing BE Degree");
   d.appendChild(t);
   document.getElementById("hh").appendChild(d);
}
</script>
</head>
<body>
<h1 onclick="s();">Welcome </h1>
<p id="hh"> CSE department </p>
<big> Governemtn college of Engineering </big>
```

```
</body>
</html>
```

Removing Elements:

```
<html>
<head>
<script type="text/javascript">
function s()
{
   document.getElementById("hh").remove();
}
</script>
</head>
<body>
<h1 onclick="s();">Welcome </h1>
<p id="hh"> CSE department </p>
<big> Governemtn college of Engineering </big>
</body>
</html>
```

Replacing one child element by another:

```
<html>
<head>
<script type="text/javascript">
function s()
{
   var d=document.createElement("p");
   var t =document.createTextNode("I am doing BE Degree");
document.getElementById("hh").parentNode.replaceChild(t,docume
nt.getElementById("hh"));
}
</script>
</head>
<body>
<h1 onclick="s();">Welcome </h1>
```

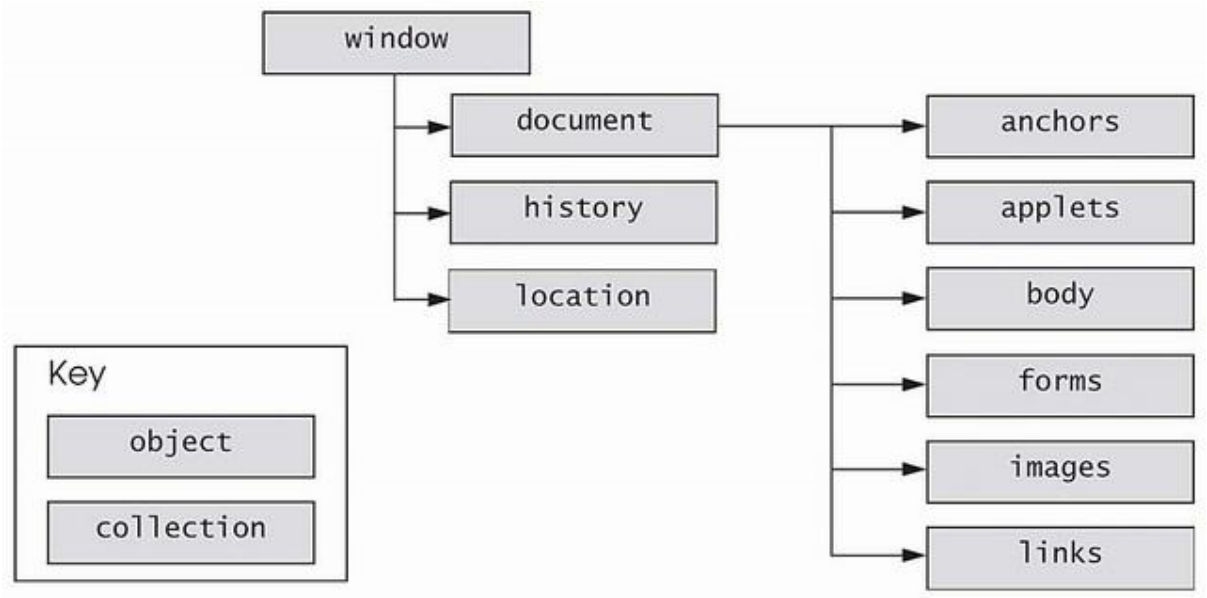
 CSE department

<big> Governemtn college of Engineering </big>

</body>

</html>

Document Object Model supported by Java Script



Regular Expression

- **Short Notation**: regex (or) RegExp
- **Definition**: It is an object which describes a pattern of characters (or) is a sequence of characters that forms a search pattern
- **Purpose**:
  1. It is used to find the formatted text string patterns from the given input string.
  2. It is used for effective and efficient text formatting and manipulations
- **Syntax for defining regular Expression**: /pattern/ modifiers
- **Modifiers used in java script:** Define global search and case sensitive

Modifiers	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

- **Range of characters prediction by using brackets**:

Range	Description
[abc]	Find the string of pattern which contain either a or b or c
[^abc]	Find the string of pattern which does not contain a or b or c
[0-9]	Find the string which contain the digit from the group 0-9
[^0-9]	Search a string which contain non-digit characters
(a/b)	Search a character either a or b

❖ **Meta characters** :

.	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: \b HI, 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*

- **Regular Expression Properties**:
  1. Global – specify if the global modifier set
  2. ignoreCase – specify if the 'i' modifier is set
  3. lastindex - The index at which to start the next match.
  4. multiline - Specifies if the "m" modifier is set.
  5. source - the source of the text pattern

- **Regular Expression methods**:
  1. **exec()** : Executes a search for a **match in a string**. It returns *an array of information or null on a mismatch*.
  2. **test()** : Tests for a match in a string. It returns true or false
- **String Methods for Regular Expression**:
  1. **Search()**: It uses an expression to search for a match, and returns the position of the match.
  2. **replace()**: It rturns a modified string where the pattern is replaced.
  3. **match()** : Returns an array containing all of the matches, including capturing groups, or null if no match is found.
  4. **matchAll()** : Returns an iterator containing all of the matches, including capturing groups.
  5. **split()**:Uses a regular expression or a fixed string to break a string into an array of substrings.

Example Program:

```
<html>
<head>
 <script type="text/javascript">
 function test()
 {
    // for search method
    var s="Welcome to cse Departments.\n CSE department is currently ruling this world";
    var n=s.search(/CSE/gi);
    document.write("Starting Match Location without case sensitive:"+n);
    n=s.search(/CSE/g);
    document.writeln("<br>Starting Match Location with case sensitive:"+n);
    n=s.search(/CSE/gmi);
    document.writeln("<br>Starting Match Location without case sensitive:"+n);

   // for replace
    var s1=s.replace(/cse/,"CSE");
    document.writeln("<br> Replaced text : "+s1);

   // for exec()
   var re=new RegExp(/([a-z]+)/i);
   var h=re.exec(s);
   if(h==null)
    document.write("<br> not exist");
   else
   {
    for(var i=0; i<h.length;i++)
       document.write("<br>"+h[i]);
   }

   //for match()
   var re=new RegExp(/([a-zA-Z0-9])/i);
```

```
    var h=s.matchAll(re);
    var i=Array.from(h);
    for(var j=0; j<i.length; i++)
       console.log("word="+ i[j]);
   }
   test();
  </script>
</head>
<body>
</body>
</html>
```

Exception Handling in Java script:

- **Exception**: Run time error
- **Types of Error**:
    1. Syntax error
    2. Logical error
    3. Latent Error
    4. Run-time error or exception

- **Keywords for handling exceptions in java script**:
    1. **Try (required)** - detect an exception in a logical block
    2. **Catch (required)** –handling the different types of exceptions (solutions for exception)
    3. **Throw** - throwing any type of exception (may be user defined or predefined)
    4. **Finally** (optional) - after try and catch, regardless of the result (it executes the statements if the case of exceptions or not in the exceptions)
- **Syntax for handling exceptions**:

```
try
{
    //main code or logic of the program
}
```

```
Catch(Exception e)
{
  //logic for handling the exception
}
…….( Optional 'N' number of catch blocks)
finally
{
  // statements need to processed in case of exception or not
}
```

- **Properties of Error Object**:
  1. name – set or return a name of the error
  2. message – set or return an error message (string format)
- **Six types of error generated by java script (Name of the error)**: (name property)
  1. **EvalError** - An error has occurred in the eval() function
  2. **RangeError**- A number "out of range" has occurred – create a error when aa numeric or parameters exceeds beyond its range
  3. **ReferenceError** - An illegal reference has occurred (variable is not declared)
  4. **SyntaxError** - A syntax error has occurred
  5. **TypeError** - A type error has occurred (unexpected types of operations)
  6. **URIError** - An error in encodeURI() has occurred (invalid characters in URI)

Example:

```
<html>  <head>  <script type="text/javascript">
function user(message)
{
  this.message = message;  this.name = 'user';
}
function hi()
{
 try
 {
   var a=parseInt(window.prompt("enter the value:"));
   var b=parseInt(window.prompt("enter the value"));
```

```
    if(b==0)
      throw new user("Divide by Zero Error");
    else if((isNaN(b) || isNaN(a)))
       throw("Not a valid input");
    else
    {
     var c=a/b;
     document.writeln("Divide = "+c);
    }
}
catch(e)
 { console.log(e);
            alert(e+" " + e.message);
             console.error();
      } }
</script>
</head>
<body >
<h1 onclick="hi();">Exception Handling </h1>
</body>
</html>
```

- **console.log()** is a function in JavaScript which is used to print any kind of variables defined before in it or to just print any message that needs to be displayed to the user.
- **Setting break points:** call "debugger;" in where ever the debugger is required
- **Falsy** Values **in Java Script**: **The following values are evaluated to false. The remaining values (including all object types) are evaluated as true**.
    1. false
    2. undefined
    3. null
    4. 0
    5. NaN
    6. the empty string ("")

Java Script Objects

- **Object based programming language**: Java script performs the various tasks using objects.
- **Basic Object terminology concepts:**
    1. **Objects** (animate objects or inanimate objects)
       Properties of Objects: attributes (value – variable) and behaviours (functions or tasks or operations)
    2. **Class** – collection of similar objects
    3. **Encapsulation**: wrapping of attributes and operations into objects
    4. **Information hiding** (object communication with another through interfaces)
    5. Abstraction
    6. Polymorphism
    7. Dynamic binding
    8. Message passing
- **Java script objects:**
    1. Math Object
    2. String Object
    3. Date Object
    4. Boolean Object
    5. Number Object

6. Document Object
7. Window object

1. Math Objects:

- ✓ **Purpose**: Perform many common mathematical calculations
- ✓ **Input**: Numbers
- ✓ **Output** : Numbers
- ✓ **Syntax:** Math.methodname() or Math.attributename
- ✓ **Math object Properties:**

S.No.	Constant	Description	Value
1.	Math.E	Base of natural logarithmic ($\log_{10}e$)	Approximately 2.718
2.	Math.LN2	Natural Logarithm of 2 ($\log_{10}2$)	Approximately 0.693
3.	Math.LN10	Natural Logarithm of 2 ($\log_{2}10$)	Approximately 2.302
4.	Math.LOG2E	Base 2 logarithm of e ($\log_{2}e$)	Approximately 1.442
5.	Math.LOG10E	Base 10 logarithm of e ($\log_{e}10$)	Approximately 0.434
6.	Math.PI	π	3.141592653589793
7.	Math.SQRT1_2	Square root of 0.5	Approximately 0.707
8.	Math.SQRT2	Square root of 2.0	Approximately 1.414

- ✓ **Math Object Functions**:

S.No.	Method	Description
1.	abs(x)	Find the absolute value of x.
2.	ceil(x)	Find the next highest integer near to x not less than x.
3.	floor(x)	Find the next lowest integer near to x not greater than x.
4.	round(x)	Round x to the closest integer
5.	lox(x)	Natural logarithmic of x (base e)
6.	exp(x)	Find the exponential value e^x.
7.	Pow(x,y)	Find the value x^y.
8.	min(x,y)	Find the minimum value of x and y
9.	max(x,y)	Find the maximum value of x and y
10.	sin(x)	Find the value trigonometric sin x value
11.	cos(x)	Find the value trigonometric cos x value
12.	tan(x)	Find the value trigonometric tan x value

2. **String Objects**:

- ✓ **Definition**: sequence of characters. *Java script supports the set of characters are called as* ***Unicode characters***.
- ✓ **Index**: Characters are accessed by using index. Index values start from 0.
- ✓ **Purpose**: Perform string operations or string manipulations
- ✓ **Input**: strings
- ✓ **Output** : Based on the type of manipulation it may be a string or number or character
- ✓ **Syntax:** Stringobjectname.methodname()
- ✓ **String Methods:**
  - **Character processing methods**: It performs string manipulations in character by character
    i. *chartAt(index)* – it returns the character in a specified index or position. (if the index is out of bounds it returns empty)
    ii. *charCodeAt(index)* – returns a Unicode character in a specified location or returns NaN if there is no character in a specified index.
    iii. *fromCharCode(value1,value2,…)* - convert the Unicode values into corresponding characters.
    Example: String.fromCharCode(64,65,66) **Output**: @AB
    iv. *toLowerCase()* – convert the strings into lowercase format
    v. *toUpperCase()* – convert the strings into upper case format
  - **Search Methods** (case sensitive)
    i. *indexOf(substring, index)* – find the starting index of the substring from the specified index. (it returns -1 if the substring is not exist) and index argument is optional. If not provided then it starts to search from 0^{th} index.
    ii. *lastIndexOf(substring, index)* – find the last occurrence starting index of the substring.
  - **Splitting and substring operations**
    i. replace(searchstring, replacestring) – replaces the first occurrence of the search string by replace string if exist. Otherwise it returns the same string without any replacement.

ii. slice(start, end) – find the substring specified form the start index to ending index (end-1). If the index is negative extraction takes place from right to left (start <end)

iii. split(delimiter) – it is used to split the string into smaller strings.

iv. substr(start, length) – find the substring from starting index

v. Substr(start, end) – find the substring from start to end.

- **Joining or concatenation or merging of string**:
  i. concat(string) – concatenate the string at the end of current source string object.
- **XHTML Markup methods**
  i. anchor(name) – wraps the source string in an anchor element (<a>..) with name as the anchor name.
  ii. link(url) – wraps the source string in anchor element with url as the hyperlink location.
  iii. Strike() – wraps the source string in <strike> element
  iv. Sub() – wraps the source string in _…element
  v. sup() – wraps the source string in ^{..}
  vi. fixed() – wraps the source string in mono spaced text format.

Example Program for String Object:

```
<html>
<head>
<script type="text/javascript">
var s="Welcome to GCE";
document.write("Sring length = "+s.length);
for(var i=0; i<s.length ; i++)
   document.write(s.charAt(i)+ "    "+s.charCodeAt(i)+"<br>");

var h=String.fromCharCode(64,65,66);
document.write("<br> string  = "+h);
var d=s.toLowerCase();
document.write("<br> lower case format = "+d);
var e=s.toUpperCase();
```

```
document.write("<br> upper case format = "+d);
document.write("<br> Index of GCE = "+s.indexOf("gce"));
document.write("<br> replace of GCE by CSE = "+s.replace("GCE","CSE"));
document.write("<br> string extraction from 0 to 4th index = "+s.slice(0,4));
document.write("<br> string extraction from -1 to -5 index = "+s.slice(-5,-1));

document.write("<br>Subscript="+"Welcome to our college".sub());
document.write("<br>Superscript="+"CSE Department".sup());
document.write("<br>fixed="+"Internet Programming" .fixed());
document.write("<br>Strike="+"I am studying in GCE" .strike());
document.write("<br>Anchor Link ="+"Click me to for further reference".link("E:/D/NKK/IP/IP/Textarea.html"));
</script>
</head><body>
<a name="cse" href="welcome.html"> clicke me  </a>
 </body>  </html>
```

Output:

Sring length = 14W 87
e 101
l 108
c 99
o 111
m 109
e 101
32
t 116
o 111
32
G 71
C 67
E 69

string = @AB
lower case format = welcome to gce
upper case format = welcome to gce
Index of GCE = -1
replace of GCE by CSE = Welcome to CSE
string extraction from 0 to 4th index = Welc
string extraction from -1 to -5 index = o GC
Subscript=Welcome to our college
Superscript=CSE Department
fixed=Internet Programming
Strike=~~I am studying in GCE~~
Anchor Link =Click me to for further reference clicke me

3. **Date Object**
   - **Purpose**: It provides a method for manipulation of date and time information.
   - **Time Zone** : Local Time Zone or Coordinated Universal Time(UTC) (Greenwich Mean Time[GMT])
   - **Syntax for creation of Date Object**: var object_name = new Date()
   - **Methods:**
     - **Reading the date and time information**:
       i. getDate() – return the day of the month
       ii. getUTCDate()
       iii. getDay() – return the day of the week 0 – Sunday , 6-Saturday
       iv. getUTCDay()
       v. getFullYear() – get the year in four digit
       vi. getUTCFullYear()

vii. getHours() – return the hours in 0 to 23 format
viii. getUTCHours()
ix. getMilliseconds()
x. getUTCMilliSeconds()
xi. getMinutes()
xii. getUTCMinutes()
xiii. getMonth()
xiv. getUTCMonth()
xv. getSeconds()
xvi. getUTCSeconds()
xvii. getTime() – returns the number of milliseconds between January 1 1970 and current time
xviii. getTimeZoneOffset()- Returns the difference in minutes between the current time on local computer and UTC

- **Setting the date and time information**:
  i. setDate(val) – set the day of the month (0 to 31)
  ii. setUTCDate(val) - set the day of the month (0 to 31)
  iii. setFullYear(y,m,d) – set the date
  iv. setUTCFullYear(y,m,d)
  v. setHours(h,m,s,ms)
  vi. setUTCHours(h,m,s,ms)
  vii. setMilliseconds(ms)
  viii. setUTCMilliSeconds(ms)
  ix. setMinutes(m,s,ms)
  x. setUTCMinutes(m,s,ms)
  xi. setMonth(m,d)
  xii. setUTCMonth(m,d)
  xiii. setSeconds(s,ms)
  xiv. setUTCSeconds(s,ms)
  xv. setTime(ms) – set the number of milliseconds between January 1 1970 and current time
- **String Conversion functions**:

i. toLocaleString() – Returns the sting representation of the date object (**Example**: 11/02/2020 10:20:22)
ii. toUTC String: Returns the string representation of the date and time ((**Example**: 11 Feb 2020 10:20:22 UTC)
iii. toString() – returns the string representation of the date and time (Tue Feb 2020 10:20:22)
iv. valueOf() – The time in milliseconds since midnight, January 1,1970. (similar to getTime())

4. **Boolean Objects**: (object wrappers)
   - **Purpose**: Manipulating Boolean values
   - **Boolean values**: true / false
   - **Syntax for creating Boolean object**:

     var object_name = new Boolean(Boolean_value)
   - **Number representation**: False – 0, null,Number.NaN or an empty String, True -1
   - **Methods**:
     i. **toString()** – returns the Boolean value as string
     ii. **valueOf()** – return true if Boolean object is true otherwise returns false

5. **Number Objects**: (object wrappers)
   - **Purpose**: It is used to represent the numerical data either in integer or floating point numbers.
   - **Primitive values**: In java script primitive values treated as objects when executing methods and properties.
   - Browser automatically converts the number literals into instance of the number class.
   - **Syntax**:

     var variable_name = new Number(value);

     Note: if non-number argument is given then it returns NaN.
   - **Properties**:

     1) MAX_VALUE – It return the maximum possible value in a number. (1.7976931348623157E+308)

2) MIN_VALUE – It return the maximum possible value in a number. (5E-324)
3) NaN – Not a number
4) NEGATIVE_INFINITY (less than the MIN_VALUE)
5) POSITIVE_INFINITY (greater than the MAX_VALUE)

- **Methods**:
  a. toExponential(number_of_digit) – It Forces a number to display in exponential notation with rounded format
  b. toFixed() – It is used to format the with the specific number of digits to the right of the decimal
  c. toLocaleString() – It returns the string value version of the current number in a fo+rmat (format is vary depends on the browser)
  d. toString() – IT returns the string representation of the number
  e. toPrecision(number_of_digits) – It represents the number of digits in right and left side of the decimal to display a number.
  f. valueOf() – It returns the primitive value of number object
  g. isNaN()
  h. isFinite()
  i. isInteger()

6. **Document Object:**
   - **Purpose**: A Document object represents the HTML document that is displayed in that window. (When an *HTML document is loaded into a web browser, it becomes a document object*)
   - The Document object has *various properties* that refer to *other objects* which allow *access to and modification* of document content.
   - *Document object is the root node* of any html document.
   - It is the property of window object.
   - **Properties**:
     a. activeElement – returns the currently focussed element in the document. (example: document.activeElement.tagName)
     b. URL – return the full URL of the HTML document
     c. title- set or return the title of the document
     d. scripts – return the collection of script elements in the document

(document.scripts[0].text, document.scripts.length, document.scripts.item(0).text)

e. anchors – return the collection of anchor elements in the document. (document.anchors.length, document.anchors[0].innerHTML)
f. body
g. document.documentElement.nodeName – return HTML
h. cookie – return the cookie information
i. documentURL – set or return the location of the document
j. images – returns the set of images in the document
k. lastModified – return the details of last modified information
l. links – return the set of links in the HTML document
m. readyState

- **Methods**:

1) addEventListener("click",function_name)
2) removeEventListener("property",function_name)
3) focus()
4) blur()
5) write() – Write the HTML expression to a document
6) writeln() – write a HTML expression to a document with newline character at the end.
7) close() – close the previously opened HTML document
8) createAttribute("attribute name") – create attributes for any HTML element

```
var att = document.createAttribute("style");
att.value = "color:red";
const h1 = document.getElementsByTagName("h1")[0];
h1.setAttributeNode(att);
```

9) createElement("element name")

```
var para = document.createElement("p");
para.innerText = "This is a paragraph.";
document.body.appendChild(para);
```

10) appendChild(textnode)
11) getElementById("Id name")

12) getElementsByClassName("class name")
13) getElementsByName("Name of the tag")
14) getElementsByTagName("tag name")

7. **Window Object:**

- It represents the open window in a browser
- **Frames Concept**: If a document contains frames (<iframe> tags), the browser creates one window object for the HTML document, and one additional window object for each frame.
- **Properties:**
  a. innerHeight – It returns the height of the window's content area (window.innerHeight).
  b. innerWidth - returns the width of the window's content area (window.innerWidth).
  c. closed – returns true or false
  d. document
  e. length – returns the number of <iframe> elements in the current window
  f. location – it returns the location of the window
  g. name – sets or return the name of the window
  h. OuterHeight – height including toolbars / scrollbars
  i. OuterWidth - width including toolbars / scrollbars
  j. pageXOffset
  k. pageYOffset
  l. parent
  m. screen -return the screen object
  n. screenX - Returns the horizontal coordinate of the window relative to the screen
  o. screen
  p. scroll
  q. scrollY

- **Methods**:
  1) Open() – open a new window
     myWindow = -*

2) window.scrollBy(x, y);
3) alert()
4) blur() - return the focus on the current window
5) focus() – creates the focus on the new window
6) print()
7) prompt()
8) clearInterval()
9) clearTimeOut()
10) confirm()
11) setInterval()
12) setTimeOut()
13) stop() – stop the window from loading
14) scrollBy(x,y) – move the scroll window in both left and down direction
15) resizeBY(x,y)

Events

- **Event:** The script which responds to user actions
- **Advantages of events:**
    1. We can create more responsive webpages
    2. Create more dynamic type of web pages
    3. Create effective interactive web pages
- **Event handlers**: The function that handles the events
- **Registering an event handler**: Assigning an event handler to an event in on a DOM tree
- **Steps for creating a webpages by using events**:
    1. Define the corresponding event handlers
    2. Registering an event handler with specific node or element in DOM tree
- **Types of Events in java script**:
    1. **Input Events**
        i. onblur – when a user leaves an input field
        ii. onchange – when a user changes the content of an input field or select a dropdown value
        iii. onfocus – when input field get focus
        iv. onselect – the input text is selected
        v. onsubmit – fires when the user clicks the submit button
        vi. onreset – fires when the user clicks reset button
        vii. onkeydown – fires when the user pressing / holding down a key
        viii. onkeypress – fires when the user pressing the key
        ix. onkeyup – fires when the user releases a key
        x. onkeydown – fires when the user pressing the key
    2. **Mouse Events**
        i. onmouseover – fires when the mouse moves over the text
        ii. onmouseout – fires when the mouse out of the particular element
        iii. onmousedown – fires when the mouse is pressed
        iv. onmouseup – fires when the mouse press is released
        v. onmousemove – fires when the mouse is moving above the image

3. **Click Events**

i. onclick – fires when the mouse is pressed either in button or any HTML element
ii. ondblclick – fires when the mouse is double clicked
iii. ondrag – fires when the element is dragged
iv. oncontextmenu – fires when the mouse is right clicked
v. ondragover – fires at the time of start operation
vi. onmousewheel – triggers when the mouse wheel is being rotated
vii. onpause - Triggers when media data is going to start playing
viii. onscroll – triggers when the elemnt scrollbar is being scrolled

4. **Load Events**

i. onload - fires when the document is loaded
ii. onerror – fires when the error is occurred while loading an image
iii. onunload – fires when the browser closes the document
iv. onresize – fires when the window browser is resized

confirm: (window.confirm("message");

- The confirm() method displays a **dialog box with a specified message, along with an OK and a Cancel button**.
- A confirm box is often used if you **want the user to verify or accept something**.
- **Note**: The confirm box takes the focus away from the current window, and forces the browser to read the message. Do not overuse this method, as it prevents the user from accessing other parts of the page until the box is closed.
- The **confirm() method returns true if the user clicked "OK", and false otherwise**.

Example:

```
var txt;    var r = confirm("Press a button!");
if (r == true) {
  txt = "You pressed OK!";
} else {
  txt = "You pressed Cancel!";    }
```

event object properties

1. screenX , screenY – it returns the coordinates of the mouse cursor on the screen coordinates system
2. clientX , client – It returns the coordinates mouse cursor inside the client area
3. altkey – this value is true if the Alt key was pressed when the event fired
4. shiftkey – this value is true if the shift key pressed when event is fired
5. type – the name of the event that fires without the prefix "on"

JSON (Java Script Object Notation)

❖ **Need for JSON**: The data exchanging between a browser and a server, the data can only be text format. Need a conversion method to convert JSON to java script object and vice versa.

❖ **Purpose of JSON**:

- JSON is a *format for storing and transporting data*.
- JSON is often used *when data is sent from a server to a web page*.

❖ **Characteristics of JSON**:

i. JSON is a lightweight data interchange format
ii. JSON is language independent *
iii. JSON is "self-describing" and easy to understand
iv. No complicated parsing and translations in JSON conversions

❖ **Code for Sending JSON Data:**

```
<html>
<head>
<script type="text/javascript">
function s()
{
  var myObj = {name: "John", age: 31, city: "New York"};
var myJSON = JSON.stringify(myObj);  // convert object into JSON
document.write(myJSON);
}
</script>
</head>
<body onclick=”s();”>
```

```
</body>
</html>
```

Receiving Data:

```
function s()
{
        var myObj = {name: "John", age: 31, city: "New York"};
        var myJSON = JSON.stringify(myObj);
        document.write(myJSON);
        var d=JSON.parse(myJSON);  // convert JSON into Java script Object
        document.write(d.name);
}
```

Code for storing JSON object in memory: localStorage.setItem("testJSON", myJSON);

Code for retrieving from memory: text = localStorage.getItem("testJSON");