Marwadi University

01CE0306-Web Technology

Unit-5 Javascript

Prof. Vaibhav K Matalia
Computer Engineering Department

Introduction to Javascript

- Javascript is a lightweight, open source and scripting language.
- JavaScript was developed by Netscape in 1995.
- It can integrate with HTML pages and enable web authors to increase interactivity of web pages.
- Everyone can use JavaScript without purchasing a license.

Use of Javascript:-

- JavaScript use to create dynamic web pages.
- JavaScript can be used for security and validations.
- JavaScript use to create richer client side interface of website(Interactive WebPages).
- JavaScript can be used to fetch information like IP address.

Java vs Javascript

Marwadi University

Java	Javascript
programming language	scripting language
It supports all the features of OOPs.	It does not support all features of OOPs.
It uses objects to perform tasks.	It is mostly function based.
Java application or programs need Java Virtual Machine(JVM),JDK and JRE to run.	Javascript needed only browser to run and need not be complied.
Java is used to develop all types of application includes desktop application as well as mobile application.	Javascript is used to make web sites and web application only.

What is script?

These are the program code written inside HTML page.

script are programs can execute on the client or server.

Difference between Client Side scripting and Server side scripting

Client side scripting	Server side scripting
Executed by a user's web browser(client),A web browser interprets the code.	Executed by the web server only when a user requests a document.
It is mainly used for client side validation which reduces number of requests that need to be passed to server.	It is usually used to provide interactive website.

It saves network bandwidth.	It does not save network bandwidth.
Cannot connect to database	Connect to the database.
Less secure because code is viewable to users through view page source command.	Secure because user is not able read source code
The scripting language needs to be enabled on the client computer.	Always enabled.
User can switch off scripting facility for security purpose.	Server side scripting cannot be blocked by user.
Example:-Javascript,Vbscript	Example:-PHP,ASP,JSP,Perl,Python

<script> tag

- To insert Javascript into an HTML page,we use the <script> tag.

<script type="text/javascript" language="javascript">
</script>

- The Javascript can be placed in the head section of your HTML or the body.

```
<html>
<body>
<script type="text/javascript">
        document.write("Marwadi university rajkot");
</script>
</body>
</html>
```

- document.write is a standard javascript command for writing output to a page.
- By entering the document.write command between the <script></script> tags, the browser will recognize it as a Javascript command and execute the code line.
- If we had not entered the script tag, browser would have treated the document.write("Marwadi university rajkot") command as pure text and just write the entire lines on the page.

External Javascript

- We can create external javascript file and embed it in many html pages.
- It provides code reusability because single javascript file can be used in several html pages.
- An external javascript file must be saved by .js extension.
- To embed the external javascript file in HTML we can use script tag with src attribute in head section.

```
<html>
<body>
        <script type="text/javascript" src="myScript.js"></script>
</body>
</html>
```

Javascript Semicolon

- In Javascript semicolons are optional. However, semicolons are required if you want to put more than one statement on a single line.

Example: semicolons are not required here

```
document.write("Web Technology")
document.write("Cyber Security")
```

Example:semicolons are required here

```
document.write("Web Technology");document.write("Cyber Security");
```

Javascript Comments

- Comments can be used to explain JavaScript code, and to make it more readable.
- JavaScript comments can also be used to prevent execution, when testing alternative code.

Single line comment:-

- Single line comments starts with //.
- Any text after // will be ignored by Javascript.

Multi-line comment:-

- Multi-line comments start with /* and end with */.
- Any text between /* and */ will be ignored by JavaScript

Note:-

White space:-Javascript ignores extra spaces.

Javascript is case sensitive:-A function named “myfunction” is not the same as “myFunction” and a variable named as “myVar” is not same as “myvar”.

Symbols:-Opening Symbols like ({ [“ ‘ , must always have a matching closing symbol,like ‘ “] })

Variables

- Variable is a container (storage area) to store data.

Example:-let num=5;

- In JavaScript, we use either var or let keyword to declare variables.
- We use the assignment operator = to assign a value to a variable.
- In JavaScript, it's possible to declare variables in a single statement.

Example:-let x=5,y=6,z=7;

- If you use a variable without initializing it, it will have an undefined value.

Example:-let x;document.write(x);

- It's possible to change the value stored in the variable.

Example:-let x=10;
x=12;
document.write(x);

Rules for Naming JavaScript Variables

- Variable names must start with either a letter, an underscore _, or the dollar sign $.
- Variable names cannot start with numbers.
- JavaScript is case-sensitive. So y and Y are different variables.
- Reserved words (like new,break,else) cannot be used as names.

Note:-

It's a good practice to give a descriptive variable name. If you are using a variable to store the number of apples, it better to use apples or numberOfApples rather than x or n.

In JavaScript, the variable names are generally written in camelCase if it has multiple words. For example, firstName, annualSalary, etc.

Note:-

Always declare JavaScript variables with var,let or const.

The var keyword is used in all JavaScript code from 1995 to 2015.

The let and const keywords were added to JavaScript in 2015.

If you want your code to run in older browsers, you must use var.

Scope of Variable

- Scope refers to the availability of variables and functions in certain parts of the code.
- In JavaScript, a variable has two types of scope:
  - Global Scope
  - Local Scope

Global scope:-

- A variable declared at the top of a program or outside of a function is considered a global variable.

Example:-

```
let x=5;
function greet()
{
        document.write(x);
}
greet();
```

- In the above program, variable x is declared at the top of a program and is a global variable. It means the variable x can be used anywhere in the program.
- The value of a global variable can be changed inside a function.

- In JavaScript, a variable can also be used without declaring it. If a variable is used without declaring it, that variable automatically becomes a global variable.

Local scope:-

- Local variables are variables that are defined within functions.
- They have local scope, which means that they can only be used within the functions that define them. Accessing them outside the function will throw an error.
- We can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.

```
let x=5;
function addition()
{
        let y=10;
        document.write(x+y);
}
addition();
document.write(x+y);//Error b is not defined
```

- In the above program, variable x is a global variable and variable y is a local variable.The variable y can be accessed only inside the function addition. Hence, when we try to access variable y outside of the function, an error occurs.

Constant

- The const keyword is used to create constants.

Example:-const PI=3.14

- Once a constant is initialized, we cannot change its value.

Example:-const PI=3.14
const PI=4.15//Error! constant cannot be changed.
const PI+10;//This will also give an error

- Simply, a constant is a type of variable whose value cannot be changed.
- We cannot declare a constant without initializing it.

Example:-const PI;//Error! Missing initializer in const declaration.
PI=3.14;

- Always declare a variable with const when you know that the value should not be changed.
- Use const when you declare regular expression or a new array.
- The const keyword is not fully supported in Internet Explorer.

Javascript Output

- Javascript can display data in different ways:
    - using document.write()
    - using innerHTML
    - using console.log()
    - using window.alert()

document.write():-

- It is a function that is used to display some strings in browser window.

Syntax:-document.write(expression1,expression2,....,expression_n)

Example:-document.write("Marwadi university rajkot")

Example:-document.write("<h1>Marwadi university rajkot</h1>")

innerHTML:-

- The innerHTML property sets or returns the HTML content (inner HTML) of an element.

```
<script>
        function f1()
        {
            document.getElementById('para1').innerHTML="Marwadi University Rajkot";
        }
    </script>
<body>
    <button onclick="f1()">CLICK HERE</button>
    <p id="para1"></p>
</body>
```

Example 2:-

```
<script>
        function f1()
        {
            var x=document.getElementById('para1').innerHTML;
            document.getElementById('para2').innerHTML=x;
            document.getElementById('para2').style.color='red';
            document.getElementById('para2').style.fontSize='24px';
        }</script>
<body>
    <p id="para1">Marwadi University Rajkot</p>
    <button onclick="f1()">CLICK HERE</button>
    <p id="para2"></p>
</body>
```

console.log():-

- All modern browsers have a web console for debugging. The console.log() method is used to write messages to these consoles.

Example:-

```
let x=55;
console.log(x);
```

- When you run the above code, 55 is printed on the console.

Syntax:-console.log(message)

- Here, the message refers to either a variable or a value.
- console.log() method is useful for testing purposes.

alert():-

- It is used to display alert box.
- It is mostly used to give a warning message to the users.
- It displays an alert dialog box that consists of some specified message (which is optional) and an OK button.
- When the dialog box pops up, we have to click "OK" to proceed.
- The alert dialog box takes the focus and forces the user to read the specified message. So, we should avoid overusing this method because it stops the user from accessing the other parts of the webpage until the box is closed.

Example 1:-Suppose we have to fill a form for an identity card. It asks about the date of birth for the eligibility criteria of the identity card. If the age is 18 years or above, then the process will continue. Otherwise, it will show a warning message that the age is below 18 years. This warning message is the 'Alert Box'.

Example 2:-Suppose a user is required to fill the form in which some mandatory fields are required to enter some text, but the user forgets to provide the input. As the part of the validation, we can use the alert dialog box to show a warning message related to fill the textfield.

Syntax:-alert(message)

- message is an string that specifies the text to display in the alert box. It consists of the information that we want to show to the users.

Example:-

```
<!DOCTYPE html>
<html>
<head>
    <script>
        function f1()
        {
            var x=document.getElementById('fullName').value;
            if(x=="")
            {
                alert('please enter your fullName');
            }
        }
    </script>
</head>
```

```
<body>
    <form>
        <input type="text" id="fullName" placeholder="Enter Your FullName"><br><br>
        <input type="button" onclick="f1()" value="CLICK HERE">
    </form>
</body>
</html>
```

popup boxes:-

- JavaScript provides the ability to pickup user input or display small amount of text to the user by using popup boxes.
- These popup boxes appear as separate windows and their content depends on the information provided by the user.
- Javascript supports three types of popup boxes.
    - alert box
    - confirm box
    - prompt box

alert box:-

- It is used when a warning message is needed to be produced.
- Other parts of the program would stop executing unless the user closes the alert box. To close the alert popup box, simply click on the ok button present in the alert box.

Syntax:-alert("Your message comes here");

Example:-

```
<!DOCTYPE html>
<html>
<head>
    <script>
    function f1()
    {
        var x=document.getElementById('txt1').value;
        var y=document.getElementById('txt2').value;
```

```
if(x=="admin" && y=="12345"){
          window.location="welcome.html";}
       else
       {alert('Your UserName and Password is incorrect');
       }
   }
   </script>
</head>
<body>
   <form>
       <input type="text" placeholder="Enter Your UserName" id="txt1"><br><br>
       <input type="password" placeholder="Enter Your Password" id="txt2"><br><br>
       <input type="button" value="VALIDATE" onclick="f1()">
   </form>
</body>
</html>
```

Confirm:-

- A confirm box is used if you want the user to verify or accept something.
- When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax:-confirm("some text")
Example:-

```
<!DOCTYPE html>
<html>
<head>
    <script>
    function f1()
    {
        var x=confirm("Are You sure you want to delete?");
```

```
if(x==true)
        {
            document.write('successfully deleted');
        }
        else
        {
            document.write('You pressed cancel');
        }
    }
    </script>
</head>
<body>
    <input type="button" value="DELETE" onclick="f1()">
</body>
</html>
```

Example:2

```
<!DOCTYPE html>
<html>
<head>
    <script>
        var x=confirm("Are You ready to visit google");
        if(x==true)
        {
            window.open("https://www.google.com");
        }
        else
        {
            document.write('You pressed cancel button');
        }
    </script>
</head></html>
```

prompt:-

- A prompt box is used if you want the user to input a value.
- When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax:-prompt("sometext","defaultText");

Example:-

```
<!DOCTYPE html>
<html>
<head>
    <script>
        let person=prompt("Enter your name","Virat Kohli");
```

```
if(person==null||person=="")
{
        document.write("User cancelled the prompt");
}
else
{
        document.write("Hello "+person);
 }
</script>
</head>
<body>
</body>
</html>
```

String

- A JavaScript string is zero or more characters written inside quotes.
- You can use single or double quotes.
- String indexes are zero-based. The first character is in position 0, the second in 1, and so on.
- Strings are immutable. That means the characters of a string cannot be changed.
- We can also create strings using the new keyword.
- Strings can be joined together with the + operator,which is called concatenation.
- It is recommended to avoid using string objects. Using string objects slows down the program.

Escape character(\):-

- It is a sequence of characters that does not represent itself when used inside a string, but is translated into another character that may be difficult or impossible to represent directly.
- If you want to add double quote in between any words, then javascript may consider it as a string. To avoid this use escape character. Following are the escape characters in JavaScript

Code	Result
\’	Single Quote
\”	Double Quote
\	Backslash
\b	Backspace
\f	Form feed
\n	New line(linux)
\r	Carriage return(Microsoft)
\t	Horizontal Tabulator
\v	Vertical Tabulator

- The 6 escape characters(\b,\f,\n,\r,\t,\v) above were originally designed to control typewriters, teletypes, and fax machines. They do not make any sense in HTML.

String Methods:-

length:-

- It returns the length of a string.
- The length property of an empty string is 0.

Syntax:-string_variable.length;

Example:-

```
let x=new String('Marwadi university');
document.write(x.length);
```

concat():-

- This method joins two or more strings.
- The concat() method does not change the existing strings. It returns a new string.

Syntax:-string_variable.concat(string1, string2, ..., stringX)

Example:-

```
var str="Javascript is a scripting is language";
var str1="Hello";
var str2="World";
console.log(str.concat(" ",str1," ",str2));//Javascript is a scripting language Hello World
```

indexOf():-

- It is used to search the position of a particular character or string in a sequence of given char values.
- This method is case-sensitive.
- The index position of first character in a string is always starts with zero. If an element is not present in a string, it returns -1.

Syntax:-
indexOf(ch):-return the index position of ch
indexOf(ch,index):-start searching the element from the provided index value and then return the index position

Example:-

```
var str="Javascript is a scripting is language";
document.write(str.indexOf("is"));//11
document.write(str.indexOf("is",17));
```

lastIndexOf():-

- It returns the index (position) of the last occurrence of a specified value in a string.
- This method searches the string from the end to the beginning.
- It returns the index from the beginning (position 0).

- It returns -1 if the value is not found.
- This method is case sensitive.

Syntax:-lastIndexOf(ch):-return the last index position of ch

lastIndexOf(ch,index):-start searching the element from the provided index value and then return the index position

Example:-

```
var str="Javascript is a scripting is language";
document.write(str.lastIndexOf("is"));//26
document.write(str.lastIndexOf("is",17));//11
```

slice():-

- The slice() method is used to fetch the part of the string and returns the new string.
- It does not change the original string.
- The start and end parameters specifies the part of the string to extract.
- The first position is 0, the second is 1.
- This method allows us to pass a negative number as an index. In such case, the method starts fetching from the end of the string.

Syntax:-string_variable(start,end)

substr():-

- The substr() method returns a portion of the string, starting at the specified index and extending for a given number of characters afterwards.
- The substr() method does not change the original string.
- To extract characters from the end of the string, use a negative start position.

Syntax:-string.substr(start, length)

start:-If start is greater than the length, substr() returns "".If start is negative, substr() counts from the end of the string.

length:-The number of characters to extract. If omitted, it extracts the rest of the string

replace():-

- It is used to replace a part of given string with some another string.
- The replace() method returns a new string.
- The replace() method does not change the original string.

Syntax:replace(searchValue, newValue)

Example:-

```
var str="Javascript is a scripting is language";
document.write(str.replace("Javascript","PHP"));
document.write(str.replace("is","are"));
document.write(str.replace(/is/g,"are"));
document.write(str.replace(/Is/gi,"are"));
```

toUpperCase():-

- It converts a string to uppercase letters.
- It does not change the original string.

Syntax:-string_variable.toUpperCase();

toLowerCase():-

- It converts a string to lowercase letters.
- It does not change the original string.

Syntax:-string_variable.toLowerCase();

trim():-

- The trim() method removes whitespace from both sides of a string.
- It does not change the original string.

Syntax:-string_variable.trim()

chatAt():-

- charAt() is a method that returns the character from the specified index.
- The index of the first character is 0, the second 1 and so on.
- It return single character.

- If no index is specified then the first character of the string is returned.

Syntax:-string_variable.charAt(index);//index must be positive number, if we specify negative index function will not work

charCodeAt():-

- This method returns the Unicode (ranging between 0 and 65535) of the character whose index is provided to the method as the argument.
- It returns NaN if the given index number is either a negative number or it is greater than or equal to the length of the string.
- The index of the first character is 0, the second is 1 and so on.

split():-

- Splits a string into array of substrings
- Return array from string

Example:-

```
var str="Marwadi university rajkot";
document.write(str.split(" "));
document.write("<br>"+str.split(" ",2));
```

Operator

- JavaScript operators are symbols that are used to perform operations on operands.
- For example 1 + 2, where + sign is an operator and 1 is left operand and 2 is right operand. + Operator adds two numeric values and produces a result which is 3 in this case.
- There are six types of operator used in JavaScript

1.Arithmetic Operator

2.Assignment Operator

3. String Operator

4. Comparison Operator

5. Logical Operator

6. Conditional Operator

Arithmetic Operator

Marwadi University

Operators	Description
*	Multiplication
/	Division
%	Modulus
+	Addition
-	Subtraction
++	Increment
--	Decrement

Assignment operator

Marwadi University

Operators	Example	Is Same as
=	x==y	x==y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

Comparison Operator

Operators	Description
<	Less than
<=	Less than equal to
>=	Greater than
==	Equality
!=	Inequality
===	Is equal to(checks for both types and value

Logical Operator

Marwadi University

Operators	Description	Example	Result
!	Logical NOT	!n1	Returns true if n1 is false
&&	Logical AND	N1>0 && n1<=100	Returns true only if both conditions are true
||	Logical OR	N1==n2 || n1==n3	Returns true if any one condition is true

Conditional Operator:-

Syntax: variablename = (condition) ? truepart : falsepart

Example: check =(name==“rajkot”) ? “OK” : “Not OK”

String Operator:-

Example:-var x=“Rajkot”+”MU”

Javascript Loops

- Loops can execute a block of code a number of times.
- JavaScript supports different kinds of loops:
  - for loop
  - for/in loop
  - for/of loop
  - while loop
  - do...while loop

for loop:-

```
for (statement 1; statement 2; statement 3)
{
  // code block to be executed
}
```

for/in loop:-

- for in statement loops through the properties of an Object.

```
for(variablename in object)
{
                    statement or block to execute
}
```

- In each iteration, one property from object is assigned to variablename and this loop continues till all the properties of the object are executed.

```
var person = {fname:"vaibhav", lname:"Patel", age:30};
for (x in person)
{
        document.write(x+":"+person[x] + "<br>");
}
```

```
var subject=["WT","CD","CS","SE"];
for (x in subject)
{
        document.write(subject[x]+"<br>");
}
```

for/of loop:-

- for of statement loops through the values of an iterable object(Array,Objects,Strings etc).

Syntax:-

```
for(variableName of Array_variable)
{
        statement or block to execute
}
```

```
var person={firstName:'Vaibhav',lastName:'Patel',age:30}
for(key of person)
{
    document.write(person[key]+"<br>");
}
```

while loop:-

- The while loop loops through a block of code as long as a specified condition is true.

Syntax:-

```
while (condition)
{
  // code block to be executed
}
```

do..while loop:-

- The do/while loop is a variant of the while loop.
- This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax:-

```
do
{
  // code block to be executed
}while (condition);
```

Conditional Statements

- JavaScript supports conditional statements which are used to perform different actions based on different conditions.
- If a condition is true, you can perform one action and if the condition is false, you can perform another action.
- There are mainly four types of conditional statements in JavaScript.

1. If Statement
2. If ... else statement
3. If ... else if.... Else statement
4. Switch statement

if statement:-

- We can use If statement if you want to check only a specific condition.
- Code is executed only if specified condition is true.

Syntax:

```
if(condition)
{
Code to be executed if condition is true
}
```

Example:-

```
var age=20;
if(age>18)
{
        document.write("<b>Qualifies for driving</b>");
}
```

Note:- if is written is lowercase letters. Using uppercase letters (IF) will generate a Javascript error!

if else statement:-

- We can use if….Else statement if you have to check two conditions and execute a different set of codes.

Syntax:

```
if(condition)
{
        Code to be executed if condition is true
}
else
{
        Code to be executed if condition is not true
}
```

Example:-

```
var age = 15;
if(age>18)
{
        document.write("<b>Qualifies for driving</b>");
}
else
{
    document.write("<b>Does not qualify for driving</b>");
}
```

if else...if else statement:-

- We can use If….Else If….Else statement if you want to check more than two conditions.

Syntax:

```
if(condition1)
{
Code to be executed if condition1 is true
}
else if(condition2)
{
Code to be executed if condition2 is true
}
else
{
Code to be executed if above all is not true
}
```

Example:-

```
var book = "maths";
if(book=="history")
{
        document.write("<b>History Book</b>");
}
else if(book=="maths")
{
    document.write("<b>Maths Book</b>");
}
else
{
    document.write("<b>Unknown Book</b>");
}
```

switch statement:-

- The JavaScript switch statement is used in decision making.
- The switch statement evaluates an expression and executes the corresponding body that matches the expression's result.

Syntax:-

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

- The switch expression is evaluated once. The value of the expression is compared with the values of each case. If there is a match, the associated block of code is executed. If there is no match, the default code block is executed.

Example:-

```
let day=new Date();
switch(day.getDay()) {
case 0:
    day = "Sunday";
    break;
case 1:
    day = "Monday";
    break;
case 2:
    day = "Tuesday";
    break;
```

```
case 3:
    day = "Wednesday";
    break;
case 4:
    day = "Thursday";
    break;
case 5:
    day = "Friday";
    break;
case  6:
    day = "Saturday";
}
document.write("Today is "+day);
```

- When JavaScript reaches a break keyword, it breaks out of the switch block.
- This will stop the execution inside the switch block.
- It is not necessary to break the last case in a switch block. The block breaks (ends) there anyway.
- If you omit the break statement, the next case will be executed even if the evaluation does not match the case.
- The default keyword specifies the code to run if there is no case match.
- If multiple cases matches a case value, the first case is selected.

User defined function

- A JavaScript function is a block of code designed to perform a particular task.
- A user-defined function saves us from rewriting the same code again and again and helps us to make our application smaller.
- Declared functions are not executed immediately. They are "saved for later use", and will be executed later, when they are invoked (called upon).
- A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ().
- Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
- Function name should be unique.

Example:-

```
<script>
function subjectList()
{
        document.write("Web Tech<br>Compiler Design<br>Cyber Security");
}
subjectList();
</script>
```

function with parameter:-

- Function parameters are the names listed in the function definition.
- Function arguments are the real values passed to the function.

Example:-

```
function myFunction(a, b)
{
 return a * b;
}
var x = myFunction(4, 3);
document.write (x);
```

default parameter values:-

- If a function is called with missing arguments(less than declared), the missing values are set to undefined.
- Sometimes this is acceptable, but sometimes it is better to assign a default value to the parameter.

Example:-

```
function myFunction(x=2,y=10)
{
  return x + y;
}
document.write(myFunction(4));
```

function expression:-

- A JavaScript function can also be defined using an expression.
- A function expression can be stored in a variable.
- After a function expression has been stored in a variable, the variable can be used as a function.

Example:-

```
const x=function (a,b){return a*b;};
let z=x(3,2);
document.write(z);
```

- The function above is actually an anonymous function (a function without a name).
- Functions stored in variables do not need function names. They are always invoked (called) using the variable name.
- The function above ends with a semicolon because it is a part of an executable statement.

Events

- The change in the state of an object is known as an Event.
- HTML events are "things" that happen to HTML elements.
- When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.
- Developers can use these events to execute JavaScript code.
- Events name begin with the letters "on".
- Events can be classified into 4 categories
    - keyboard events
    - mouse events
    - form events
    - mouse events

Marwadi University

Keyboard events:-

- Whenever a user presses any key on the Keyboard, different events are fired.
- There are three keyboard events, namely keydown, keypress, and keyup.
- One of the most common uses of keyboard events today is computer gaming. Most browser-based games require some form of keyboard input.

onkeydown:-

- The onkeydown event occurs when the user presses a key on the keyboard.

Syntax:-<element onkeydown="myScript">

- You can write onkeydown in all HTML element except <head>, <html>, <iframe>, <meta>, <script>, <style>, and <title>

onkeypress:-

- The onkeypress event occurs when the user presses a key on the keyboard.

Syntax:-<element onkeydown="myScript">

- You can write onkeypress in all HTML element except <head>, <html>, <iframe>, <meta>, <script>, <style>, and <title>

Example:-

```
<script>
function f1(event)
{
var x=event.keyCode;
document.getElementById("demo").innerHTML=x;
}
</script>
```

```
<body>
<input type=“text” onkeypress="f1(event)"/>
<p id="demo"></p>
</body>
```

Note:-The onkeypress event is deprecated. It is not fired for all keys (like ALT, CTRL, SHIFT, ESC) in all browsers. To detect if the user presses a key, always use the onkeydown event. It works for all keys.

onkeyup:-

- The onkeyup event occurs when the user releases a key on the keyboard.

Syntax:-<element onkeyup="myScript">

- You can write onkeyup in all HTML element except
,<head>, <html>, <iframe>, <meta>, <script>, <style>, and <title>

Example 1:-

```
<script language="javascript">
function f1()
{
var d=document.getElementById("nm1");
nm2.value=d.value;
}
</script>
Enter string<input type="text" id="nm1"onkeyup="f1()"><br>
copy string<input type="text" id="nm2">
```

Example 2:-

```
<html>
  <head>
    <script>
        function sayHello() {
          var str = document.getElementById("subject");
          str.value = str.value.toLowerCase();
        }
    </script>
  </head>
  <body>
    <input type = "text" onkeyup = "sayHello()" id = "subject">
  </body>
</html>
```

Mouse events:-

onclick():-

- It occurs when the user clicks on an HTML element.

Syntax:-<element onclick="myScript">

- You can write onclick in all HTML element except
, <head>, <html>, <iframe>, <meta>,<script>, <style>, and <title>

Example:-

```
<!DOCTYPE html>
<html>
<head>
    <script>
        function autoFillAddress()
        {
            var x=document.getElementById('addressLine1');
            var y=document.getElementById('addressLine2');
            var checkBox=document.getElementById('chckbox');
```

```
if(checkBox.checked==true){y.value=x.value;}
            else
            {y.value='';}
        }
    </script>
</head>
<body>
    <form>
        Current Address <textarea rows="7" cols="25"
id="addressLine1"></textarea><br><br>
        <input type="checkbox" id="chckbox" onclick="autoFillAddress()"> Same as
permanent address<br></br>
        Permanent address<textarea rows="7" cols="25" id="addressLine2"></textarea>
    </form></body></html>
```

ondblclick():-

- It occurs when the user double-clicks on an HTML element.

Syntax:-<element ondblclick="myScript">

- You can write ondblclick in all HTML element except
, <head>, <html>, <iframe>, <meta>,<script>, <style>, and <title>

```
<script>
    function f1()
    {
        document.getElementById('para1').innerHTML='Marwadi University Rajkot';
    }
</script>
<body>
            <input type="button" ondblclick="f1()" value="Double click to see effects">
            <p id="para1"></p>
</body>
```

onmouseover():-

- The onmouseover event in JavaScript gets activated when the mouse cursor moves over an HTML element.
- The onmouseover event does not activate when the user clicks an element.
- For example, an onmouseover event can be used to highlight a hyperlink whenever the user hovers the mouse over it.

onmouseenter():-

- The onmouseenter event occurs when the mouse pointer enters an element.
- The onmouseenter event is often used together with the onmouseleave event, which occurs when the mouse pointer leaves an element.

onmousemove():-

- It occurs when the pointer moves over an element.

Note:-difference between mouseover(),mouseenter() and mousemove()

mouseover:-The onmouseover event triggers when the mouse pointer enters an element or any one of its child elements.

mouseenter:-The onmouseenter event is triggered only when the mouse pointer hits the element.

mousemove:-The onmousemove event is triggered each time the mouse pointer is moved when it is over an element.

```
<!DOCTYPE html>
<html>
<style>
div {
  width: 150px;
  height: 100px;
  border: 1px solid black;
  margin: 10px;
  padding: 5px;
  text-align: center;
  background-color: lightgray;
}
h3
{
    background-color: rgb(228, 15, 200);
}
</style>
```

```
<body>
<h2>onmousemove</h2>
<div onmousemove="myMoveFunction()">
  <h3 id="demo1">Mouse over me!</h3>
</div>
<h2>onmouseenter</h2>
<div onmouseenter="myEnterFunction()">
  <h3 id="demo2">Mouse over me!</h3>
</div>
<h2>onmouseover</h2>
<div onmouseover="myOverFunction()">
  <h3 id="demo3">Mouse over me!</h3>
</div>
```

```
<script>
let x = 0;
let y = 0;
let z = 0;
function myMoveFunction() {
  document.getElementById("demo1").innerHTML = z+=1;
}
function myEnterFunction() {
  document.getElementById("demo2").innerHTML = x+=1;
}
function myOverFunction() {
  document.getElementById("demo3").innerHTML = y+=1;
}
</script>
</body>
</html>
```

onmouseout():-

- The onmouseout event occurs when the mouse pointer moves out of an element.
- The onmouseout event is often used together with the onmouseover event, which occurs when the pointer is moved over an element.

onmouseleave():-

- It occurs when the mouse pointer leaves an element.
- The onmouseleave event is often used together with the onmouseenter event, which occurs when the mouse pointer enters an element.

```
<!DOCTYPE html>
<html>
    <head>
        <style>
        div
        {
            width: 100px;
            height: 100px;
            border: 1px solid black;
            margin: 10px;
            float: left;
            padding: 30px;
            text-align: center;
            background-color: lightgray;
        }
```

```
        p
        {
            background-color: white;
        }
        </style>
</head>
<body>
<div onmousemove="myMoveFunction()">
    <p>onmousemove: <br> <span id="demo">Mouse over and leave me!</span></p>
</div>
<div onmouseleave="myLeaveFunction()">
    <p>onmouseleave: <br> <span id="demo2">Mouse over and leave me!</span></p>
</div>
<div onmouseout="myOutFunction()">
    <p>onmouseout: <br> <span id="demo3">Mouse over and leave me!</span></p>
</div>
```

```
<script>
var x = 0,y = 0,z = 0;
function myMoveFunction() {
    document.getElementById("demo").innerHTML = z+=1;
}
function myLeaveFunction() {
    document.getElementById("demo2").innerHTML = x+=1;
}
function myOutFunction() {
    document.getElementById("demo3").innerHTML = y+=1;
}
</script>
</body>
</html>
```

Note:-The mouseleave event only occurs when the mouse pointer is moved out of the div element.The onmouseout event occurs when the mouse pointer is moved out of the div element, and when it leaves its child elements (p and span).

onmousedown():-The onmousedown event occurs when a user presses a mouse button over an HTML element.

onmouseup():-The onmouseup event occurs when a mouse button is released over an element.

```
<body>
    <p id="para1" onmouseup="f2()" onmousedown="f1()">Lorem ipsum dolor </p>
    <script>
        function f1()
        {
            document.getElementById('para1').style.color='red';
        }
        function f2()
        {
            document.getElementById('para1').style.color='green';
        }
    </script>
</body>
```

Form events:-

onchange():-

- The onchange event attribute works when the value of the element changes and select the new value from the List.
- You can write onchange with following HTML Tags:<input type="checkbox">, <input type="file">, <input type="password">, <input type="radio">, <input type="range">, <input type="search">, <input type="text">, <select> and <textarea>

Example:-

```
<script>
        function test1()
        {
                var d=document.getElementById("opt").value;
                document.getElementById("a1").innerHTML=d;
        }
</script>
```

```
<body>
        <form>
                <select id="opt" onchange="test1()">
                        <option value="">Select course</option>
                        <option>Engineering</option>
                        <option>MCA</option>
                        <option>MBA</option>
                </select>
        <h1>select stream is=<b id="a1"></h1>
        </form>
</body>
```

onfocus():-

- The onfocus attribute fires the moment that the element gets focus.
- onfocus is most often used with <input>, <select>, and <a>.

onblur():-

- The onblur attribute fires the moment that the element loses focus.
- onblur is most often used with form validation code (e.g. when the user leaves a form field).

Example:-

```
<form>
        <input type="text" id="hh1" placeholder="Enter Your FirstName" onfocus="f1()"
            onblur="f2()"><span id="hh2"></span><br><br>
        <input type="text">
 </form>
 <script>
        function f1()
        {
            document.getElementById('hh1').style.background='lightgrey';
        }
```

```
function f2()
        {
            var x=document.getElementById('hh1').value;
            var firstNameCheck=/^[a-z]+$/;
            if(firstNameCheck.test(x))
            {
                document.getElementById('hh2').innerHTML='';
            }
            else
            {
                document.getElementById('hh2').innerHTML='Invalid';
            }
        }
    </script>
</body>
```

onsubmit():-

- The onsubmit attribute fires when a form is submitted.
- You can write onsubmit event with form attribute only.

onreset():-

- The onreset attribute fires when a form is reset.
- You can write onreset event with form attribute only.

```
<form onreset="f2()" onsubmit="f1()">
        Enter name: <input type="text"><br><br>
        <input type="submit"> <input type="reset">
</form>
<script>
    function f1()
    {
      alert("The form was submitted");
    }
    function f2() {
      alert("The form was reset");
    }
</script>
</body>
```

windows events

onload():-

- The onload attribute fires when an object has been loaded.
- onload is most often used within the <body> element to execute a script once a web page has completely loaded all content (including images, script files, CSS files, etc.). However, it can be used on other elements as well.
- The onload attribute can be used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.
- You can write onload event with following HTML tags:<body>,<frame>, <frameset>, <iframe>, <img>, <input type="image">, <link>, <script> and <style>

Example:-

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
  alert("Page is loaded");
}
</script>
</head>
<body onload="myFunction()">
<h1>Marwadi University Rajkot</h1>
</body>
</html>
```

onunload():-

- The onunload attribute fires once a page has unloaded (or the browser window has been closed).
- onunload occurs when the user navigates away from the page (by clicking on a link, submitting a form, closing the browser window, etc.)
- Due to different browser settings, this event may not always work as expected.

Example:-

```
<!DOCTYPE html>
<html>
<body onunload="myFunction()">
<script>
function myFunction() {
  alert("Thank you for visiting W3Schools!");
}</script>
```

onresize():-

- The onresize attribute fires when the browser window is resized.
- You can write onresize event with body tag only.

Example:-

```
<body onresize="myFunction()">
<script>
function myFunction() {
  alert("You have changed the size of the browser window!");
}
</script>
</body>
```

onerror():-

- The onerror attribute fires when an error occurs while loading an external file (e.g. a document or an image).
- You can write onerror event with following HTML tags:<img>, <input type="image">, <link>, and <script>

Example:-

```
<body>
<img src="1.jpg" onerror="myFunction()">
<script>
function myFunction() {
  alert("The image could not be loaded.");
}
</script>
</body>
```

DOM

- It is an API for manipulating HTML elements.(add,remove and modify elements)
- When a web page is loaded, the browser creates a Document Object Model of the page.
- In this DOM tree, the document is the root node or object.
- With object model
  Change all the HTML elements in page
  change all the HTML attributes in page
  change all the CSS styles in the page
  remove existing HTML elements and attributes
  add new HTML elements and attributes
  react to all existing HTML events in the page
- In the DOM, all HTML elements are defined as objects.so it will have both property and method.
- If you want to access any HTML element in an HTML page, you always start with accessing the document object.

DOM Methods

Marwadi University

getElementsByClassName():-

- It is used for selecting or getting the elements through their class name value.
- This DOM method returns an array-like object that consists of all the elements having the specified classname.
- On calling the getElementsByClassName() method on any particular element, it will search the whole document and will return only those elements which match the specified or given class name.

Example:-

```
<body>
        <ul>
            <li class="subject">Web Technology</li>
            <li>Compiler Design</li>
            <li class="subject">Cyber Security</li>
        </ul>
```

```
<script>
        var x=document.getElementsByClassName('subject');
        for(var i=0;i<x.length;i++)
        {
            x[i].style.color='red';
        }
</script>
</body>
```

getElementsByTagName():-It returns the collection of all elements in the document with given tag name.

Example:-

```
<body>
        <h2>Javascript</h2>
        <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Sunt, ducimus?</p>
        <h2>PHP</h2>
        <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit.</p>
        <script>
            var x=document.getElementsByTagName('h2');
            for(var i=0;i<x.length;i++)
            {
                x[i].style.color='red';
            }
       </script>
</body>
```

getElementsByName():-

- It returns a collection of elements with specified name.
- This collection is called node list and each element of the node list can be visited with the help of the index.

Example:-

```
<body>
    <form>
        <input type="radio" name="gender" value="male">Male
        <input type="radio" name="gender" value="male">Female
        <span id="hh1"></span><br><br>
        <input type="button" onclick="f1()" value="CLICK HERE">
    </form>
```

```
<script>
        function f1()
        {
            var gen=document.getElementsByName('gender');
            if(gen[0].checked==true||gen[1].checked==true)
            {
                document.getElementById('hh1').innerHTML='';
            }
            else
            {
                document.getElementById('hh1').innerHTML='select your gender';
            }
        }
    </script>
</body>
```

getElementById():-

- It returns the element of specified id.
- It returns null if the element does not exist.
- It is one of the most common methods in the HTML DOM. It is used almost every time you want to read or edit an HTML element.

Example:-

```
<script type="text/javascript">
function getcube(){
var number=document.getElementById("number").value;
alert(number*number*number);
}
</script>
```

```
<form>
Enter No:<input type="text" id="number" name="number"/><br/>
<input type="button" value="cube" onclick="getcube()"/>
</form>
```

write():-

- It is a function that is used to display some strings in the output of HTML web pages (Browser window).

Syntax:-document.writeln(exp1,exp2,...,expn)

- By using document.write() method, we can pass an expression called as 'single argument' or can pass multiple values separated by commas "," known as 'multiple arguments'.
- These values can be displayed in sequential order as they are written in the method of document.write in javascript.

Array

- JavaScript array is a single variable that is used to store different elements.
- It is often used when we want to store a list of elements and access them by a single variable.

Create an Array

- You can create an array using two ways:

1.Using an array literal:-The easiest way to create an array is by using an array literal [].
Example:-const sem6=["WT","CD","CS","SE"];

2.Using the new keyword

- You can also create an array using JavaScript's new keyword.

const sem6=new Array("WT","CD","CS","SE");

- It is recommended to use array literal to create an array.
- Here, sem6 is an array. The array is storing 4 values.

Access Elements of an Array

- You can access elements of an array using indices (0, 1, 2 …). For example,

```
const sem6=["WT","CD","CS","SE"];
console.log(sem6[0]);//WT
console.log(sem6[2]);//CS
```

Looping Array Elements

- One way to loop through an array, is using a for loop:

```
const sem6=["WT","CD","CS","SE"];
for(i=0;i<sem6.length;i++)
{
        document.write(sem6[i]+"<br>");
}
```

Array Methods

length:-

- The length property return length of an array.

```
const sem6=["WT","CD","CS","SE"];
document.write("arr.length is : " +sem6.length);
```

push():-

- The push() method adds one or more elements to the end of an array.

```
var a=["vaibhav","harsh"];
a.push("jill","Divyesh");
document.write(a);
```

unshift():-

- It is used to add one or more elements to the beginning of the given array.
- This method accepts a single parameter. This parameter element is to be added at the beginning of the array.

```
const sem6=["WT","CD","CS","SE"];
sem6.unshift("Python","Computer Network");
document.write(sem6);
```

pop():-

- It is used to remove the last element of the array and also returns the removed element.
- This method does not accept any parameter.
- If the array is empty, then this function returns undefined.

```
const sem6=["WT","CD","CS","SE"];
sem6.pop();
document.write(sem6);
```

shift():-

- It removes the first element of the array.
- This method does not accept any parameter.
- If the array is empty then this function returns undefined.

```
const sem6=["WT","CD","CS","SE"];
sem6.shift();
document.write(sem6);
```

concat():-

- The concat() method is used to merge two or more arrays together.

```
var a=["vaibhav","raj"];
var b=["jill","divyesh"];
var c=a.concat(b);
document.write(c);
```

sort():-

- The sort() method will sort the elements alphabetically.

```
var arr=["AngularJS","Nodejs","JQuery","Bootstrap"]
arr.sort();
document.writeln(arr); //AngularJS,Bootstrap,JQuery,Nodejs
var priceList = [1000, 50, 2, 7, 14];
priceList.sort();
document.writeln(priceList);
```

reverse():-

- The reverse() method returns the array in reverse order.
- The reverse() method does not take any parameters.

```
let numbers = [1, 2, 3, 4, 5];
numbers.reverse();
document.write(numbers);
```

Window Scrolling

scrollTo():-

- The scrollTo() method scrolls the document to specified coordinates.
- For the scrollTo() method to work, the document must be larger than the screen, and the scrollbar must be visible.

Syntax:-window.scrollTo(x,y) OR scrollTo(x,y)

Parameters:-

x:-Required.The coordinate to scroll to (horizontally), in pixels.

y:-Required.The coordinate to scroll to (vertically), in pixels.

scroll():-

- The Window scroll() method scrolls the window to a particular place in the document. This method is different form scrollTo() method as it takes different parameters like scrolling behavior, etc using ScrolltoOptions Dictionary.

Syntax:-window.scroll(x-coord, y-coord) OR window.scroll(top,left,behavior)

Parameters:-

top:-Specifies the number of pixels along the Y axis to scroll the window or element.

left:-Specifies the number of pixels along the X axis to scroll the window or element.

behavior:-Determines whether scrolling is instant or animates smoothly. This option is a string which must take one of the following values:

smooth:-scrolling should animate smoothly

instant:-scrolling should happen instantly in a single jump

```
<body>
	<script language="JavaScript">
	function function1(){
		window.scrollTo(200,200);
	}
	function function2(){
		window.scroll(400,400);
	}
	</script>
	<input type="button" value="Scroll to (200,200)" onClick="function1();">
	<input type="button" value="Scroll(400,400)" onClick="function2();">
	<div style="height:1000;width:1500;background-color:blue;"></div>
</body>
```

Marwadi University

Thank You