JAVASCRIPT

JavaScript is used in Web pages to add functionality, validate forms, communicate with the server, and much more. A scripting language is a lightweight programming language. JavaScript is programming code that can be inserted into HTML pages.

<SCRIPT> TAG:
To insert a JavaScript into an HTML page, use the <script> tag. The <script> and </script> tells where the JavaScript starts and ends. The lines between the <script> and </script> contain the JavaScript.

Example:
<head>
<script type=”text/javascript”>
document.write (“Hello World”);
</script>
</head>

LOCATION FOR JAVASCRIPT CODE:

There are three general areas that JavaScript can be placed for use in a webpage.

- Inside the head tag.
- Within the body tag of the document.
- In an External file.

EXTERNAL JAVASCRIPTS:
Scripts can also be placed in external files. External files often contain code to be used by several different web pages. External JavaScript files have the file extension .js. To use an external script, point to the .js file in the "src" attribute of the <script> tag.

Example:

< script src =”script1.js” ></ script >

Manipulating JavaScript Elements:

To access an HTML element from JavaScript, you can use the document.getElementById(*id*) method. Use the "id" attribute to identify the HTML element:

Comments in JavaScript:

Comments will not be executed by JavaScript. Comments can be added to explain the JavaScript, or to make the code more readable. Single line comments start with //.

EVENTS IN JAVASCRIPT:

Events are signals generated when specific actions occur. JavaScript is aware of these signals, and scripts can be built to react to these events.
There are several events in JavaScript like:

- Page Events
- Form Events
- Keyboard Events
- Mouse Events

Page Events:

Load	The onLoad event fires when the page has completely finished loading, including all graphics and external files
Resize	The onResize event fires when the page is resized.
Scroll	The onScroll event fires when the page is scrolled. The event can also be attached to other objects that are scrollable, such as a text area.
Unload	The onUnload event fires when the user leaves the page, either by closing the window/tab, clicking a link, hitting the back button, or otherwise navigating to a different page.

Form Events:

Blur	The onBlur event fires whenever a form field *lose* focus.
Change	The onChange event fires whenever the value of a form field (including select lists) changes.
Copy	The onCopy event fires whenever a form field's contents are copied
Cut	The onCut event fires whenever a form field's contents are cut
Focus	The onFocus event fires whenever a form field gains focus, such as with a mouse click inside it or tabbing from another field.
Keydown	The onKeydown event fires when a key is pushed down while the target field has focus.
Keyup	The onKeyup event fires when a key is release while the target field has focus.
Keypress	The onKeypress event fires when a key is pressed and at an interval determined by the operating system as long as the key is still held down.
Paste	The onPaste event fires whenever a form field's contents are pasted
Reset	**T**he onReset event fires when a form's reset button is pushed
Select	The onSelect event fires when a field's text is highlighted (selected).
Submit	The onSubmit event fires when a form's submit button is pushed, or if the form is otherwise submitted (such as when the user presses the enter key).

Keyboard Events:

Copy, cut, paste	The onCopy, onCut, onPaste event fires whenever a form field's contents are copied, cut, pasted respectively with the user's local copy function
Keydown	The onKeydown event fires when a key is pushed down while the target field has focus
Keypress	The onKeypress event fires when a key is pressed and at an interval determined by the operating system as long as the key is still held down.
Keyup	The onKeyup event fires when a key is release while the target field has focus.

Mouse Events:

Click	The onClick event fires whenever a mouse button is clicked
Contextmenu	The onContextmenu event fires whenever the right mouse button is clicked
Dblclick	The onDblClick event fires whenever a mouse button is clicked twice within a time frame determine by the operating system (a double- click).
Mousedown	The onMousedown event fires whenever a mouse button is pressed down.
Mousemove	The onMousemove event fires whenever a mouse is moved.
Mouseout	The onMouseout event fires whenever a mouse moves outside of the bounds of its target object after being inside those bounds. It only fires once, but will re-fire if the mouse moves back inside and then outside again.
Mouseover	The onMouseover event fires whenever a mouse moves inside of the bounds of its target object after being outside those bounds. It only fires once, but will re-fire if the mouse moves back outside and then inside again.
Mouseup	The onMouseup event fires whenever a mouse button is released.
Right click	This event fires whenever the right mouse button is clicked
Scrolling	This event fires when the page is scrolled

STATEMENTS IN JAVASCRIPT

JavaScript statements are "commands" to the browser. The purpose of the statements is:

- To tell the browser what to do.
- Can set a variable equal to a value.
- Can call a function.

Categories of Statements:

- Conditional Statements
- Loop Statements
- Object Manipulation Statements
- Comment Statements
- Exception Handling Statements

Conditional Statements:

The conditional statement is a statement that will evaluate to be either true or false. The most common type of conditional statements used checks to see if something equals a value.
Example:

```
var num= 8;
if(num==7)
{
        document.write("Lucky no. 7");
}
else
{
        Document.write("You are unlucky today!!")
}
```

Loop Statements:

A loop statement checks to see if some condition is true, and if taht condition is true, it executes a chunk of code. After that code is executed, the condition is checked again. If it is true, the process starts over again; if it is false, the loop stops and the rest of the code continue along.

Example:

```
<script type="text/javascript">
for(i=0;  i<5; i++)
{
        document.write ("Counter i=" + i);
}
```

Comment Statement:

Comments will not be executed by JavaScript. Comments can be added to explain the JavaScript, or to make the code more readable. Single line comments start with //.

Object Manipulation Statement:

These are statements that are used to take advantage of the object model to get tasks done.

JAVASCRIPT ALERT:

The JavaScript alert is a dialogue box that pops up and takes the focus away from the current window and forces the web browser to read the message.

Example:

```
<form>
<input type= ”button” onClick= “alert(‘Do you really want to leave this page??’)”
        Value= “Yes”>
</form>
```

JAVASCRIPT CONFIRM:

The JavaScript *confirm* function is very similar to the JavaScript *alert* function. A small dialogue box pops up and appears in front of the web page currently in focus. The confirm box is different from the alert box. It supplies the user with a choice; they can either press OK to confirm the pop up’s message or they can press cancel and not agree to the popup’s request.

Example:

```
<script type=”text/ javascript”>
function confirmation()
{
        var answer= confirm(“Leave this Web page??”)
        if (answer)
        {
        alert (“Have a nice day!!”);
        window.location=http://www.google.com;
        }
        else
        {
        alert (“Thanks for sticking around!!”);
        }
}
```

JAVASCRIPT PROMPT:

It is used to gather the information to be displayed in an alert dialog box.

Example:

```
<head><script type=”text/javascript”>
function prompter( )
{
        var show= prompt (“Hey!! What is your name??”);
        alert (“Nice to see you around” + show + ”!!” );
}
</script></head>
<body>
<input type=”button” onClick=”prompter( )” value=”Say my name”>
</body>
```

JAVASCRIPT PRINT SCRIPT:

It performs the same operation as the print option in “File” menu of the browser.

Example:

```
<form>
<input type=”button” value=”Print this page!!” onClick=window.print( )”/>
</form>
```

JAVASCRIPT REDIRECT:

If you want to redirect to some other page window.location is used.

Example:

```
<script type=”text/javascript”>
window.location=”www.google.co.in”;
</script>
```

JAVASCRIPT DATE OBJECT:

The Date object is useful when you want to display a date or use a timestamp in some sort of calculation. If no argument is supllied to the Date constructor, then it will create a Date object based on the visitor’s internal clock.

Example:

```
<script type=”text/javascript”>
var currenttime= new Date( );
</script>
```

FUNCTIONS IN JAVASCRIPT

NUMBER METHODS:

Method	Description
constructor()	Returns the function that created this object's instance. By default this is the Number object.
toExponential()	Forces a number to display in exponential notation, even if the number is in the range in which JavaScript normally uses standard notation.
toFixed()	Formats a number with a specific number of digits to the right of the decimal.
toLocaleString()	Returns a string value version of the current number in a format that may vary according to a browser's locale settings.
toPrecision()	Defines how many total digits (including digits to the left and right of the decimal) to display of a number.
toString()	Returns the string representation of the number's value.
valueOf()	Returns the number's value.

BOOLEAN METHODS:

Method	Description
toSource()	Returns a string containing the source of the Boolean object; you can use this string to create an equivalent object.
toString()	Returns a string of either "true" or "false" depending upon the value of the object.
valueOf()	Returns the primitive value of the Boolean object.

STRING HTML WRAPPERS:

Method	Description
anchor()	Creates an HTML anchor that is used as a hypertext target.
big()	Creates a string to be displayed in a big font as if it were in a <big> tag.
blink()	Creates a string to blink as if it were in a <blink> tag.
bold()	Creates a string to be displayed as bold as if it were in a <b> tag.
fixed()	Causes a string to be displayed in fixed-pitch font as if it were in a <tt> tag
fontcolor()	Causes a string to be displayed in the specified color as if it were in a <font color="color"> tag.
fontsize()	Causes a string to be displayed in the specified font size as if it were in a <font size="size"> tag.
italics()	Causes a string to be italic, as if it were in an <i> tag.
link()	Creates an HTML hypertext link that requests another URL.
small()	Causes a string to be displayed in a small font, as if it were in a <small> tag.
strike()	Causes a string to be displayed as struck-out text, as if it were in a <strike> tag.
sub()	Causes a string to be displayed as a subscript, as if it were in a <sub> tag
sup()	Causes a string to be displayed as a superscript, as if it were in a <sup> tag

ARRAY METHODS:

Method	Description
concat()	Returns a new array comprised of this array joined with other array(s) and/or value(s).
every()	Returns true if every element in this array satisfies the provided testing function.
filter()	Creates a new array with all of the elements of this array for which the provided filtering function returns true.
forEach()	Calls a function for each element in the array.
indexOf()	Returns the first (least) index of an element within the array equal to the specified value, or -1 if none is found.
join()	Joins all elements of an array into a string.
lastIndexOf()	Returns the last (greatest) index of an element within the array equal to the specified value, or -1 if none is found.
pop()	Removes the last element from an array and returns that element.
push()	Adds one or more elements to the end of an array and returns the new length of the array.
reduce()	Apply a function simultaneously against two values of the array as to reduce it to a single value.
reduceRight()	Apply a function simultaneously against two values of the array as to reduce it to a single value.
reverse()	Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first.
shift()	Removes the first element from an array and returns that element.
slice()	Extracts a section of an array and returns a new array.
some()	Returns true if at least one element in this array satisfies the provided testing function.
toSource()	Represents the source code of an object
sort()	Sorts the elements of an array.
splice()	Adds and/or removes elements from an array.
toString()	Returns a string representing the array and its elements.
unshift()	Adds one or more elements to the front of an array and returns the new length of the array.

MATH METHODS:

Method	Description
abs()	Returns the absolute value of a number.
acos()	Returns the arccosine (in radians) of a number.
asin()	Returns the arcsine (in radians) of a number.
atan()	Returns the arctangent (in radians) of a number.
atan2()	Returns the arctangent of the quotient of its arguments.
ceil()	Returns the smallest integer greater than or equal to a number.
cos()	Returns the cosine of a number.

floor()	Returns the largest integer less than or equal to a number.
log()	Returns the natural logarithm (base E) of a number.
max()	Returns the largest of zero or more numbers.
min()	Returns the smallest of zero or more numbers.
pow()	Returns base to the exponent power, that is, base exponent.
random()	Returns a pseudo-random number between 0 and 1.
round()	Returns the value of a number rounded to the nearest integer.
sin()	Returns the sine of a number.
sqrt()	Returns the square root of a number.
tan()	Returns the tangent of a number.
toSource()	Returns the string "Math".

STRING METHODS:

Method	Description
charAt()	Returns the character at the specified index.
charCodeAt()	Returns a number indicating the Unicode value of the character at the given index.
concat()	Combines the text of two strings and returns a new string.
indexOf()	Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found.
lastIndexOf()	Returns the index within the calling String object of the last occurrence of the specified value, or -1 if not found.
localeCompare()	Returns a number indicating whether a reference string comes before or after or is the same as the given string in sort order.
length()	Returns the length of the string.
match()	Used to match a regular expression against a string.
replace()	Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring.
search()	Executes the search for a match between a regular expression and a specified string.
slice()	Extracts a section of a string and returns a new string.
split()	Splits a String object into an array of strings by separating the string into substrings.
substr()	Returns the characters in a string beginning at the specified location through the specified number of characters.
substring()	Returns the characters in a string between two indexes into the string.
toLocaleLowerCase()	The characters within a string are converted to lower case while respecting the current locale.
toLocaleUpperCase()	The characters within a string are converted to upper case while respecting the current locale.
toLowerCase()	Returns the calling string value converted to lower case.
toString()	Returns a string representing the specified object.
toUpperCase()	Returns the calling string value converted to uppercase.
valueOf()	Returns the primitive value of the specified object.

DATE METHODS:

Method	Description
Date()	Returns today's date and time
getDate()	Returns the day of the month for the specified date according to local time.
getDay()	Returns the day of the week for the specified date according to local time.
getFullYear()	Returns the year of the specified date according to local time.
getHours()	Returns the hour in the specified date according to local time.
getMilliseconds()	Returns the milliseconds in the specified date according to local time.
getMinutes()	Returns the minutes in the specified date according to local time.
getMonth()	Returns the month in the specified date according to local time.
getSeconds()	Returns the seconds in the specified date according to local time.
getTime()	Returns the numeric value of the specified date as the number of milliseconds since January 1, 1970, 00:00:00 UTC.
getTimezoneOffset()	Returns the time-zone offset in minutes for the current locale.
getUTCDate()	Returns the day (date) of the month in the specified date according to universal time.
getUTCDay()	Returns the day of the week in the specified date according to universal time.
getUTCFullYear()	Returns the year in the specified date according to universal time.
getUTCHours()	Returns the hours in the specified date according to universal time.
getUTCMinutes()	Returns the minutes in the specified date according to universal time.
getUTCMonth()	Returns the month in the specified date according to universal time.
getUTCSeconds()	Returns the seconds in the specified date according to universal time.
getYear()	**Deprecated** - Returns the year in the specified date according to local time. Use getFullYear instead.
setDate()	Sets the day of the month for a specified date according to local time.
setFullYear()	Sets the full year for a specified date according to local time.
setHours()	Sets the hours for a specified date according to local time.
setMilliseconds()	Sets the milliseconds for a specified date according to local time.
setMinutes()	Sets the minutes for a specified date according to local time.
setMonth()	Sets the month for a specified date according to local time.
setSeconds()	Sets the seconds for a specified date according to local time.
setTime()	Sets the Date object to the time represented by a number of milliseconds since January 1, 1970, 00:00:00 UTC.
setUTCDate()	Sets the day of the month for a specified date according to universal time.
setUTCFullYear()	Sets the full year for a specified date according to universal time.
setUTCHours()	Sets the hour for a specified date according to universal time.
setUTCMilliseconds()	Sets the milliseconds for a specified date according to universal time.
setUTCMinutes()	Sets the minutes for a specified date according to universal time.

setUTCMonth()	Sets the month for a specified date according to universal time.
setUTCSeconds()	Sets the seconds for a specified date according to universal time.
setYear()	**Deprecated** - Sets the year for a specified date according to local time. Use setFullYear instead.
toDateString()	Returns the "date" portion of the Date as a human-readable string.
toGMTString()	**Deprecated** - Converts a date to a string, using the Internet GMT conventions. Use toUTCString instead.
toLocaleDateString()	Returns the "date" portion of the Date as a string, using the current locale's conventions.
toLocaleFormat()	Converts a date to a string, using a format string.
toLocaleString()	Converts a date to a string, using the current locale's conventions.
toLocaleTimeString()	Returns the "time" portion of the Date as a string, using the current locale's conventions.
toSource()	Returns a string representing the source for an equivalent Date object; you can use this value to create a new object.
toString()	Returns a string representing the specified Date object.
toTimeString()	Returns the "time" portion of the Date as a human-readable string.
toUTCString()	Converts a date to a string, using the universal time convention.
valueOf()	Returns the primitive value of a Date object.

VALIDATIONS IN JAVASCRIPT

Userid:
- must be of length 5 to 12

```
function userid_validation(uid,mx,my)
{
    var uid_len = uid.value.length;
    if (uid_len == 0 || uid_len >= my || uid_len < mx)
    {
        alert("User Id should not be empty / length be between "+mx+" to "+my);
        uid.focus();
        return false;
    }
    return true;
}
```

Password:
- must be of length 7 to 12

```
function passid_validation(passid,mx,my)
{
    var passid_len = passid.value.length;
    if (passid_len == 0 ||passid_len >= my || passid_len < mx)
    {
        alert("Password should not be empty / length be between "+mx+" to "+my);
        passid.focus();
        return false;
    }
    return true;
}
```

User name:
- alphabets only

```
function allLetter(uname)
{
    var letters = /^[A-Za-z]+$/;
    if(uname.value.match(letters))
        {   return true;  }
    else
    {
    alert('Username must have alphabet characters only');
    uname.focus();  return false;
    }
}
```

Email validation:

```
function ValidateEmail(uemail)
{
    var mailformat = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
    if(uemail.value.match(mailformat))
    {
        return true;
    }
    else
    {
        alert("You have entered an invalid email address!");
        uemail.focus();
        return false;
    }
}
```

Radio buttons:

- suppose we have two fields male and female

```
function validgender(umale,ufem)
{
    x=0;
    if(umale.checked)
    {
        x++;
    }
    if(ufem.checked)
    {
        x++;
    }
    if(x==0)
    {
        alert('Select Male/Female');
        umale.focus();
        return false;
    }
    else
    {
        alert('Form Successfully Submitted');
        window.location.reload()
        return true;
    }
}
```

Zip code or phone no.:
- numeric only

```
function allnumeric(uzip)
{
    var numbers = /^[0-9]+$/;
    if(uzip.value.match(numbers))
    {
        return true;
    }
    else
    {
        alert('ZIP code must have numeric characters only');
        uzip.focus();
        return false;
    }
}
```