2026

JAVA Script

DR. RAJESH KUMAR OJHA

DR. R. K. OJHA

1. Introduction to JavaScript

- JavaScript is a programming language that can be included on web pages to make them more interactive.
- You can use it to check or modify the contents of forms, change images, open new windows and write dynamic page content.
- JavaScript is a very powerful client-side scripting language.
- JavaScript is used mainly for enhancing the interaction of a user with the webpage. In other words, you can make your webpage livelier and more interactive, with the help of JavaScript.
- JavaScript is also being used widely in game development and mobile application development.
- JavaScript was developed by Brendan Eich in 1995, which appeared in Netscape, a popular browser of that time.

Features of JS:-

1. client-side
2. interpreted
3. object oriented
4. high level language
5. dynamic
6. easy to write

Advantages of JS:-

- **Interpreted languages**: JavaScript is an interpreted language. It requires no compilation process so no compiler is required. The browser interprets JavaScript as it HTML tags.
- **Easy to learn**: The syntax of JavaScript is very easy. Any person can learn it very easily and use it to develop dynamic and attractive websites.
- **Easy to Debug and Test**: JavaScript code is interpreted line by line. The errors are indicated along with line number. It is very easy to find error in the code, correct it and test it gain.

- **Event-Based Programming**: JavaScript is an event-based language. It means that different code segment is executed when certain event occur. For example, a code segment may execute when the user clicka button or moves a mouse over an object etc.
- **Procedural Capabilities**: JavaScript provides all capabilities of a procedural language. It provides condition checking, loops and branching facilities that can be executed in a web page.
- **Platform Independence**: JavaScript is platform independent language. Any JavaScript-enabled browser can understand and interpreted JavaScript code. Any JavaScript code is executed on different types of hardware.

Disadvantages of JS:-

- **Security Issues**: Java script snippets, once appended onto web pages execute on client system immediately and therefore can also be used to exploit the user's system. While a certain restriction is set by modern web standards on browsers, malicious code can still be executed complying with the restrictions set.
- **Java-script rendering varies**: Different layout engines may render Java script differently resulting in inconsistency in terms of functionality and interface. While the latest versions of java script and rendering have been geared towards a universal standard, certain variations still exist.

Uses of JS:-

- **Input Validation**: JavaScript can be used to validate the input. Data entered informs should be validated before it is processed.
- **Mouse Rollover Effects**: JavaScript can used to create different buttons with interesting mouse rollover effects. It makes browsing more interesting and attractive.
- **Popup Windows**: JavaScript can be used to create popup windows. These windows are normally used to display important announcements, offers and news etc.
- **Dynamic Contents**: JavaScript can be used to generate dynamic contents in a website. Different HTML tags can be generated based on the user input etc.
- **User Interaction**: JavaScript can be used to interest with the user. The input entered by the user can be processed and proper message can be displayed to the user. The interactive capabilities of a website makes it more interesting and productive for the users.

2. Elements of JavaScript

Character-set:-

It combines small alphabets,capital alphabets,digits and special symbols to write a program.

- a-z(ascii code->97-122)
- a-z(ascii code->65-90)
- 0-9(ascii code->48-57)

Constants:-

Constants are the fixed quantity whose value can never be change throughout the program execution.

Types:-

1. Integer constant-

 -may be +ve or –ve.

 -examples are 9,-346

2. Float/real constant-

 -may be +ve or _ve

 -examples are 1/3,-4.89

3. Character/string constant-

 -these constants are enclosed within a pair of double quote and size is not fixed. ex-“anil”, ”s”

Variables:-

These are the temporary named memory locations where the constant values are stored. Every constant value must be associated with a variable inside the memory.

Rules for constructing a variable: -

1. variable name must start with an alphabet or an underscore (_).
2. it cannot start with a digit.
3. but digits can be placed after first place.

4. no blank space is allowed while declaring a variable.
5. variable name length must not be more than 38 characters.
6. no special symbols except underscore is allowed while declaring a variable.

Comment Lines:-

These are used to deactivate line of code(s) which are present but not used inside the program.

Types:

1. single line comment: -used to deactivate a single line using the symbol (//).

2. multiline comment: -used to deactivate more than one line at a time.(/* */)

Datatypes:-

Datatype decides which type of dada a variable can take. it always used as a suffix for the variable. There are different types datatypes used in c programming language.

System requirement to write a program:-

1. an editor:-notepad/notepad++/editplus/dreamweaver
2. a browser:- google chrome/mozila/opera/interner explorer

how to write an run java-script programs:-

1. first of all open an editor like notepad and write your program/code.
2. Then save this file just like html means choose any location and give a file name with extension ".html" and save the file.
3. Go to that location and double click on the file to view the output.

(1)Write a simple program to print your name ?

```
<html>
<head>
<title>abc</title>
```

```
<script>
document.write(“hello students”);
</script>
</head>
<body bgcolor=”pink” text=”violet”>
</body>
</html>
```

(or)

```
<html>
<head>
<title>abc</title>
</head>
<body bgcolor=”pink” text=”violet”>
<script>
document.write(“hello students”);
</script>
</body>
</html>
```

Description:-

“document” is a pre-defined object in java-script which is used to call a pre-defined method.

“write” is a method/function in java-script which is used to print a message on the screen.

Operators:-

These are the special symbols which are used to operate or perform task over the operand or variable.

Types:-

1. **Assignment operator:-** used to assign a value to the variable. it is of two types.
    i. simple assignment(=)

ex: var a=5;

ii. compound assignment(+=,-=,*=,/=)

ex: sum+=i;

2. **Special operator: -**

(i)comma: -used to separate constants, variable and expressions.

Ex:-

constant separation: -var a=new Array(1,2,3,4,5);

variable separation: -var a,b,c;

expression separation: -var a=5,b=4;

(ii)semicolon: - used to terminate/close a line of code.

Ex:- var a=9;

3. **Concatenation operator**: - (+)used to concatenate/join string with variable part or with tags.

Ex:- document.write("addition value="+c);

4. **Arithmetic operator:-** used to perform mathematical operation over operands using different types of symbols like +,-,*,/,%.

(4)Write a program to add two integer number ?

```
<html>
<head>
<title>abc</title>
<script>
var a=5,b=4,c;
c=a+b;
document.write("addition value="+c);
</script>
</head>
<body bgcolor="pink" text="violet">
```

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

(5)Write a program to input two integer number and find their addition, subtraction, multiplication, division and modulo-division ?

```
<html>
<head>
<title>abc</title>
<script>
var n1=10,n2=2,a,s,m,d,md;
a=n1+n2;
s=n1-n2;
m=n1*n2;
d=n1/n2;
md=n1%n2;
document.write(“addition value=”+a+”<br>”);
document.write(“subtraction value=”+s+”<br>”);
document.write(“multiplication value=”+m+”<br>”);
document.write(“division value=”+d+”<br>”);
document.write(“modulo division value=”+md);
</script>
</head>
<body bgcolor=”pink” text=”violet”>
</body>
</html>
```

(7)Write a program to find the area and circumference of a circle ?

```
<html>
```

```
<head>
<title>abc</title>
<script>
var r=6,a,c;
a=3.141*r*r;
c=2*3.141*r;
document.write(“area of circle=”+a+”<br>”);
document.write(“circumference of circle=”+c);
</script>
</head>
<body bgcolor=”pink” text=”violet”>
</body>
</html>
```

(8)Write a program to input temperature in farhenheit scale and display it in celsius scale ?

```
<html>
<head>
<title>abc</title>
<script>
var f=98.7,c;
c=(f-32)*5/9;
document.write(“temperature in celsius scale is=”+c);
</script>
</head>
<body bgcolor=”pink” text=”violet”>
</body>
</html>
```

(9)Write a program to swap two integer number using third variable ?

```
<html>
<head>
<title>abc</title>
<script>
var a=5,b=4,c;
c=a;
a=b;
b=c;
document.write(“after swapping value of a=”+a+”<br>”);
document.write(“after swapping value of b=”+b);
</script>
</head>
<body bgcolor=”pink” text=”violet”>
</body>
</html>
```

(10)Write a program to swap two integer number without using third variable ?

```
<html>
<head>
<title>abc</title>
<script>
var a=8,b=7;
a=a+b;
b=a-b;
a=a-b;
document.write(“after swapping value of a=”+a+”<br>”);
document.write(“after swapping value of b=”+b);
</script>
</head>
<body bgcolor=”pink” text=”violet”>
</body>
</html>
```

(12)Write a program to calculate simple interest ?

```
<html>
<head>
<title>abc</title>
<script>
var p=5000,r=2,t=3;
si=(p*r*t)/100;
document.write(“simple interest=”+si);
</script>
</head>
<body bgcolor=”pink” text=”violet”>
</body>
</html>
```

5. **Relational/Comparison operator:-** used to compare between two or more operands using different symbols like >,<,>=,<=,==,!=.
   ex:- a>b
6. **Logical/Boolean operator:-** used to perform logical operation like producing true/false value using and,or,not,xor gates.

<table>
<tr><th>AND(&&)</th><th>OR(||)</th><th>NOT(!)</th><th>XOR(^)</th></tr>
<tr><td>T T->T</td><td>T T->T</td><td>T->F</td><td>T T->F</td></tr>
</table>

T	F->F	T	F->T	F->T	T	F->T
F	T->F	F	T->T		F	T->T
F	F->F	F	F->F		F	F->F

7. **Unary operator:-** the operator which is used to perform over a single operand is called unary operator.
   ex:- a++
8. **Binary operator:-** the operator which is use to operate over two operand is called binary operator.
   ex:- a+b
9. **Ternary/conditional operator:-** the operator which is used to operate over more than two operands are called ternary operator.

syn:- (condition)? print statement1:print statement2;

(13)Write a program to find the greater number among two inputted number using ternary operator ?

```
<html>
<head>
<title>abc</title>
<script>
var a=5,b=6;
(a>b)?document.write(“a is greater”):document.write(“b is greater”);
</script>
</head>
<body bgcolor=”pink” text=”violet”>
</body>
</html>
```

(14)Write a program to check for leap year using ternary operator ?

```
<html>
<head>
<title>abc</title>
```

```
<script>
var y=2018;
(((y%100!=0)&&(y%4==0))||(y%400==0))?document.write(“leap year”):document.write(“not”);
</script>
</head>
<body bgcolor=”pink” text=”violet”>
</body>
</html>
```

10. **Bitwise operator:-**the operator which is used to perform bit operation over the operand or variable as 0 and 1 is called bitwise operator.

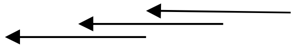
    types:-

    i. bitwise and(&)
    ii. bitwise or(|)
    iii. bitwise xor(^)
    iv. bitwise complement(~)
    v. bitwise left shift(<<)
    vi. bitwise right shift(>>)

(16)Write a program for bitwise left-shift operator ?

```
<html>
<head>
<title>abc</title>
<script>
var a=7,b=2,c;
c=a<<b;
document.write(“value of  c=”+c);
</script>
```

```
</head>
<body bgcolor=”red” text=”green”>
</body>
</html>
```



0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0

$(11100)_2=1x2^4+1x2^3+1x2^2+0x2^1+0x2^0=16+8+4+0+0=28$

(17)Write a program for bitwise complement operator ?

```
<html>
<head>
<title>abc</title>
<script>
var a=5,b;
b=~a;
document.write(“value of b=”+b);
</script>
</head>
<body bgcolor=”red” text=”green”>
</body>
</html>
```

(18)Write a program for bitwise and operator ?

```
<html>
<head>
<title>abc</title>
<script>
var a=14, b=7,c;
```

```
c=a&b;
document.write(“value of  c=”+c);
</script>
</head>
<body bgcolor=”red” text=”green”>
</body>
</html>
```

11.Increment and decrement operator:-

i. Increment operator:- this operator is used to increase the value of the variable by one in two different ways.

   a. pre-increment:- first increase the value after that update it.
      ex- ++a

   b. post-increment:- first update the value after increase it.
      ex- a++

(19)Write a program for increment operator ?

```
<html>
<head>
<title>abc</title>
<script>
var a=5,b,c;
b=a++  +   ++a;
c=++a  +  a++;
document.write(“value of  b=”+b+”<br>”);
document.write(“value of  c=”+c);
</script>
</head>
```

```
<body bgcolor=”red” text=”green”>
</body>
</html>
```

ii. Decrement operator:- this operator is used to decrease the value of the variable by one in two different ways.

a. pre-decrement:- first increase the value after that update it.
   ex- --a

b. post-decrement:- first update the value after increase it.
   ex- a--

(20)Write a program for decrement operator ?

```
<html>
<head>
<title>abc</title>
<script>
var a=5,b,c;
b=a--;
c=a--;
document.write(“value of  b=”+b+”<br>”);
document.write(“value of  c=”+c);
</script>
</head>
<body bgcolor=”red” text=”green”>
</body>
</html>
```

3. Control Structure

Control structure is the control flow of program execution where we can use different types of statements to process our programs. There are different types of statements available in control structures that are:

i. conditional statements
ii. unconditional statements
iii. looping statements

i. Conditional statements:- These are the types of statements where we use conditions to process the program using some relational operators to produce true or false value.

Types:-

1. **If/simple if:-**
   Here we are able to specify only one condition and one print statement. The demerit of this statement is, it cannot go to the default part.

```
syn:- if(condition)
        print statement;
```

(21)Write a program to check whether an inputted number is even or not using simple if ?

```
<html>
<head>
<title>abc</title>
<script language=”JavaScript”>
var a=8;
if(a%2==0)
document.write(“number is even”);
</script>
</head>
<body bgcolor=”cyan” text=”red”>
```

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

2. **If else:-**

   We are able to specify only one condition but two print statement means one for true part and another for false part.

```
syn:- if(condition)
          print statement1;
         else
          print statement2;
```

(22)Write a program to input two integer number and find the greater one using if else ?

```
<html>
<head>
<title>series</title>
<script language=”JavaScript”>
var a=3,b=4;
if(a>b)
document.write(“a is greater”);
else
document.write(“b is greater”);
</script>
</head>
<body bgcolor=”gray” text=”indigo”>
</body>
</html>
```

3. **Nested if:-**

Here we are able to specify ‘n’ number of conditions and ‘n’ number of print statements. But the problem is that it is a very complex type of statement where we get confuse about the starting and ending of if block because of nested structure.

```
Syn:- if(condition-1)
        if(condition-2)
           print statement1;
        else
           print statement2;
                .
                .
       else
        if(condition-n)
           print statement-1;
        else
           print statement-n;
```

(23)Write a program to input a year and check whether it is leap year or not using nested if ?

```
<html>
<head>
<title>series</title>
<script language=”JavaScript”>
var y=2018;
if(y%100!=0)
if(y%4==0)
document.write(“leap year”);
else
document.write(“not”);
```

```
else
if(y%400==0)
document.write(“leap year”);
else
document.write(“not”);
</script>
</head>
<body bgcolor=”green” text=”red”>
</body>
</html>
```

(24)Write a program to input three integer number and find the greatest one using nested if ?

```
<html>
<head>
<title>series</title>
<script language=”JavaScript”>
var a=45,b=23,c=90;
if(a>b)
if(a>c)
document.write(“a is greatest”);
else
document.write(“c is greatest”);
else
if(b>c)
document.write(“b is greatest”);
else
document.write(“c is greatest”);
```

```
</script>
</head>
<body bgcolor=”green” text=”red”>
</body>
</html>
```

4. **Ladder else if:-**

   It is the type of conditional statement where we can specify ‘n’ number of conditions and ‘n’ number of print statements. It is very simple as compare to nested if because here no nesting of condition is there and we can specify one condition and one print statement so on.

```
syn:- if(condition-1)
         print statement-1;
else if(condition-2)
print statement-2;
                  .
                  .
else
print statement-n;
```

(25)Write a program to input an integer number and check whether it is even or odd and positive or negative using ladder else if ?

```
<html>
<head>
<title>series</title>
<script language=”JavaScript”>
var n=5;
if(n%2==0&&n>0)
document.write(“number is even and positive”);
```

```
else if(n%2!=0&&n<0)
document.write(“number is odd and negative”);
else if(n%2==0&&n<0)
document.write(“number is even and negative”);
else if(n%2!=0&&n>0)
document.write(“number is odd and positive”);
else
document.write(“special number”);
</script>
</head>
<body bgcolor=”green” text=”red”>
</body>
</html>
```

(26)write a program to input three integer number and find the greatest one using ladder else if ?

```
<html>
<head>
<title>series</title>
<script language=”JavaScript”>
var a=5,b=90,c=67;
if(a>b&&a>c)
document.write(“a is greatest”);
else if(b>a&&b>c)
document.write(“b is greatest”);
else
document.write(“c is greatest”);
</script>
```

```
</head>
<body bgcolor=”green” text=”red”>
</body>
</html>
```

(28)Write a program to find the grade of an emp by finding gross salary where basic,ta,da,hra are given using ladder else if ?

```
<html>
<head>
<title>series</title>
<script language=”JavaScript”>
var bs=24500,ta,da,hra,gs;
ta=0.05*bs;
da=0.075*bs;
hra=0.1*bs;
gs=bs+ta+da+hra;
if(gs>=100000)
document.write(“a grade employee”);
else if(gs>=75000&&gs<100000)
document.write(“b grade employee”);
else if(gs>=50000&&gs<75000)
document.write(“c grade employee”);
else if(gs>=20000&&<50000)
document.write(“d grade employee”);
else
document.write(“e grade employee”);
</script>
```

```
</head>
<body bgcolor=”green” text=”red”>
</body>
</html>
```

(29)Write a program to find the daily wages of a worker according to the following conditions using ladder else if statement?

duty in hours	**amount in rupees**
within first 8 hours	100 rupees
next 4 hours	20 rs/hr
next 4 hours	40 rs/hr
next 4 hours	60 rs/hr
next 4 hours	80 rs/hr

```
<html>
<head>
<title>series</title>
<script language=”JavaScript”>
var hr=17,amt;
if(hr>=1&&hr<=8)
amt=100;
else if(hr>=9&&hr<=12)
amt=100+(hr-8)*20;
else if(hr>=13&&hr<=16)
amt=180+(hr-12)*40;
else if(hr>=17&&hr<=20)
amt=340+(hr-16)*60;
else if(hr>=21&&hr<=24)
```

```
amt=580+(hr-20)*80;
document.write(“amount incurred by the worker=”+amt);
</script>
</head>
<body bgcolor=”green” text=”red”>
</body>
</html>
```

ii. **Unconditional statements:-** These are the types of statements where we don’t use any condition rather we use no. of cases to produce true or false statement. Cases consist of constants or expressions.

Types:-

1. **Switch-case statement:-**

It is the type of unconditional statement where we are able to specify n no of cases. Case blocks are created by the help of keyword case and exited by the help the keyword break. If no case is satisfied inside the switch block then it goes to the default block specified by default keyword.

```
syn:- switch(expression)
      {
            case <case-constant 1>:
            statement(s);
            break;
            .
            .
            case <case-constant n>:
            statement(s);
            break;
            default:
            statement(s);
```

```
          }
```

(31)Write a program to display the day name using switch case ?

```
<html>
<head>
<title>series</title>
<script language=”JavaScript”>
var day=1;
switch(day)
{
case 1:
document.write(“the day is sunday”);
break;
case 2:
document.write(“the day is monday”);
break;
case 3:
document.write(“the day is tuesday”);
break;
case 4:
document.write(“the day is wednesday”);
break;
case 5:
document.write(“the day is thursday”);
break;
case 6:
document.write(“the day is friday”);
break;
case 7:
document.write(“the day is saturday”);
break;
default:
document.write(“wrong choice”);
}
</script>
</head>
<body bgcolor=”green” text=”red”>
</body>
</html>
```

(32)Write a program to input a character and check whether it is vowel or consonant using switch case ?

```
<html>
<head>
<title>series</title>
<script language=”JavaScript”>
var ch=”u”;
switch(ch)
{
case ‘a’:
case ‘e’:
case ‘i’:
case ‘o’:
case ‘u’:
case ‘A’:
```

```
case ‘E’:
case ‘I’:
case ‘O’:
case ‘U:
document.write(“it is a vowel”);
break;
default:
document.write(“it is consonant”);
}
</script>
</head>
<body bgcolor=”green” text=”red”>
</body>
</html>
```

(33)Write a program to implement a calculator using switch case ?

```
<html>
<head>
<title>series</title>
<script language=”JavaScript”>
var a=10,b=2,c;
var choice=”*”;
switch(choice)
{
case ‘+’:
c=a+b;
break;
case ‘-‘:
c=a-b;
break;
case ‘*’:
c=a*b;
break;
case ‘/’:
c=a/b;
break;
case ‘%’:
c=a%b;
break;
default:
document.write(“wrong choice”);
}
document.write(“calculated value=”+c);
</script>
</head>
<body bgcolor=”green” text=”red”>
</body>
</html>
```

iii. **Looping statements:-** These are also called iterative or repeatative statements. When we want to execute or print one statement for more than one time with a single specification then we use the concept of looping. For creating a looping statement we need components.

a. **Initialization**:- from which the variable value starts execution or printing.

b. **Condition**:- upto which the variable value should goes.

c. **Incr/decr**:- used to increase or decrease the value of the variable after each execution.

Types:-

1. **While loop:-**

   It is otherwise called as top-tested loop or pre-tested loop or entry control loop. In this type of looping statement first the condition is checked after the statements are get executed and printed. if the condition is false then no statement is executed or printed.

```
syn:- variable value initialization;
          while(condition)
          {
                    statement(s);
                    incr/decr;
          }
          print statement(s);
```

Wap to print 1 to 10 using while loop ?

```
<html>
<head>
<title>series</title>
<script language=”JavaScript”>
var i=1;
while(i<=10)
{
document.write(“<h1>”+“value of i=”+i+”</h1>”);
i++;
}
</script>
</head>
```

```
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

Wap to input a number and check whether it is prime or not using while loop ?

```
<html>
<head>
<title>series</title>
<script language=”JavaScript”>
var i=1,n=7,c=0;
while(i<=n)
{
if(n%i==0)
c++;
}
if(c==2)
document.write(“<h1>”+“number is prime” +”</h1>”);
else
document.write(“<h1>”+“number is not prime” +”</h1>”);
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

Wap to input a decimal number and find its binary equivalent using while loop ?

```
<html>
```

```
<head>
<title>series</title>
<script language=”JavaScript”>
var num=5,prd=1,sum=0,rem;
while(num!=0)
{
rem=num%2;
sum=sum+rem*prd;
prd=prd*10;
num=num/2;
}
document.write(sum);
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

2. **Do-while loop:-**

   It is otherwise called as bottom-tested loop or post-tested loop or exit control loop. in this type of looping statement first the statements are get executed and printed after that the condition is checked. if the condition is false then no statement is executed or printed.

```
syn:- variable value initialization;
        do
        {
                statement(s);
                incr/decr;
```

```
          } while(condition);
          print statement(s);
```

Wap to print the series like 11,22,...,99 using do while loop ?

```
<html>
<head>
<title>series</title>
<script language=”JavaScript”>
var i=11;
do
{
document.write(“<h1>”+“value of  i=”+i+”</h1>”);
i=i+11;
}
while(i<=99);
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

Wap to input a number and reverse it using do while loop ?

```
<html>
<head>
<title>series</title>
<script language=”JavaScript”>
var num=123,rev=0,rem;
do
```

```
{
rem=num%!0;
rev=rev*10+rem;
num=num/10;
}
while(num|=0);
document.write("reverse value="+rev);
</script>
</head>
<body bgcolor="red" text="blue">
</body>
</html>
```

3. For loop:-

It is the simplest type of looping statement because here all the three parts of the loop written in one line; so it reduce the line of codes.

syn:- for(initialization; condition; incr/decr)

Wap to print the series like 50,45,...,5 using for loop?

```
<html>
<head>
<title>series</title>
<script language="JavaScript">
var i;
for(i=50;i>=5;i=i-5)
{
document.write("value of i="+i);
}
```

```
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

Wap to find the sum of all even and product of all odd numbers between 1 to 50 using for loop ?

```
<html>
<head>
<title>series</title>
<script language=”JavaScript”>
var i,sum=0,prd=1;
for(i=1;i<=10;i++)
{
if(i%2==0)
sum=sum+i;
else
prd=prd*i;
}
document.write(“value of sum=”+sum+”<br>”);
document.write(“value of prd=”+prd);
}
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

Wap to print Fibonacci series like 0,1,1,2,3,5,8,13,21,34 using for loop ?

```
<html>
<head>
<title>series</title>
<script language=”JavaScript”>
var i,a=0,b=1,sum=0;
document.write(a+”<br>”);
document.write(b+”<br>”);
for(i=1;i<=8;i++)
{
sum=a+b;
document.write(c+”<br>”);
a=b;
b=sum;
}
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

Wap to find hcf and lcm of two numbers using for loop ?

```
<html>
<head>
<title>series</title>
<script language=”JavaScript”>
var i,hcf,lcm,a=10,b=20;
```

```
for(i=1;i<=a&&i<=b;i++)
{
if(a%i==0&&b%i==0)
hcf=i;
}
lcm=(a*b)/hcf;
document.write("value of hcf="+hcf+"<br>");
document.write("value of lcm="+lcm);
</script>
</head>
<body bgcolor="red" text="blue">
</body>
</html>
```

Nested for:-

When one or more than one for loops are nested inside another for loop then it is called nested for loop.

(55)Write a program for pyramid series ?

```
<html>
<head>
<title>abc</title>
</head>
<script language="JavaScript">
var i,j;
for(i=1;i<=5;i++)
{
for(j=1;j<=i;j++)
document.write(j);
```

```
1

1  2

1  2  3

1  2  3  4

1  2  3  4  5
```

```
document.write ("<br>");
}
</script>
</head>
<body>
</body>
</html>
```

(56)Write a program for pyramid series ?

```
<html>
<head>
<title>abc</title>
<script>
for(i=5;i>=1;i--)
{
for(j=1;j<=i;j++)
document.write(j);
document.write ("<br>");
}
</script>
</head>
<body>
</body>
</html>
```

```
1    2    3    4    5

1    2    3    4

1    2     3

1    2

1
```

(58)Write a program for pyramid series ?

```
<html>
<head>
```

```
<title>abc</title>
<script type=”JavaScript”>
var i,j,k=1;
for(i=1;i<=5;i++,k=k+2)
{
for(j=i;j<=4;j++)
document.write(“&nbsp&nbsp&nbsp“);
for(j=1;j<=k;j++)
document.write(“*”);
document.write (“<br>”);
}
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

```
                *
            *   *   *
        *   *   *   *   *
    *   *   *   *   *   *   *
*   *   *   *   *   *   *   *   *
```

(59)Write a program to print floyd triangle upto five rows ?

```
<html>
<head>
<title>abc</title>
</script lang=”JavaScript”>
var i,j,k=1;
for(i=1;i<=5;i++)
{
for(j=1;j<=i;j++)
{
```

```
1

2   3

4   5   6

7   8   9   10

11  12  13  14  15
```

```
document.write(k);
k++;
}
document.write (“<br>”);
}
</script>
</head>
<body>
</body>
</html>
```

4. Array

Array is a derived data-type in java-script. It is also an object in java-script. It is used to store homogeneous/similar type of data. Whenever we want to store more than one value of similar type inside a single variable using a subscript/size then we use the concept of array. Every element should be separated by a comma and are accessed by the help of a unique index number which is by default start from zero.

Syn:-

```
var <variablename>=new Array(values);
```

note:- values are optional here.

Write a program to print array elements ?

```
<html>
<head>
<title>array</title>
<script language=”JavaScript”>
var a=new Array(3,66,12,70,5);
document.write(“first element=”+a[0]+”<br>”);
document.write(“second element=”+a[1]+”<br>”);
document.write(“third element=”+a[2]+”<br>”);
document.write(“fourth element=”+a[3]+”<br>”);
document.write(“fifth element=”+a[4]);
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

Write a program to print array elements ?

```
<html>
<head>
<title>array</title>
<script language=”JavaScript”>
var a=new Array(3,66,12,70,5);
var i;
for(i=0;i<5;i++)
{
document.write(a[i]+”<br>”);
}
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

Write a program to print array elements in reverse order?

```
<html>
<head>
<title>array</title>
<script language=”JavaScript”>
var a=new Array(3,66,12,70,5);
var i;
for(i=4;i>=0;i--)
{
document.write(a[i]+”<br>”);
}
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

Pre-defined methods of array:-

1. reverse():-used to reverse all the elements.

Ex:-

```
<html>
<head>
<title>array</title>
<script language=”JavaScript”>
var a=new Array(3,66,12,70,5);
var b=a.reverse();
document.write(b);
</script>
```

```
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

o/p-5,70,12,66,3

2. sort():- used to sort array elements in ascending order by default.

Ex:-

```
<html>
<head>
<title>array</title>
<script language=”JavaScript”>
var a=new Array(“r”,”e”,”k”,”a”,”s”);
var b=a.sort();
document.write(b);
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

o/p-a,e,k,r,s

3.push():- used to push element at the last index of the array.

Ex:-

```
<html>
<head>
<title>array</title>
<script language=”JavaScript”>
```

```
var a=new Array(9,33,12,90,6);
var b=a.push(55);
document.write(a);
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

o/p-9,33,12,90,6,55

4.pop():- used to delete last element from the last index of the array.

Ex:-

```
<html>
<head>
<title>array</title>
<script language=”JavaScript”>
var a=new Array(9,33,12,90,6);
var b=a.pop();
document.write(a);
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

o/p-9,33,12,90

5.join():- used to join array elements to create a string using delimeter.

Ex:-

```
<html>
<head>
<title>array</title>
<script language=”JavaScript”>
var a=new Array(9,33,12,90,6);
var b=a.join(“*”);
document.write(b);
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

o/p-9*33*12*90*6

6.splice():- used to removes or replaces or adds elements by specifying position and number of items.

Ex:-

```
<html>
<head>
<title>array</title>
<script language=”JavaScript”>
var a=new Array(9,33,12,67);
var b=a.splice(2,0,89,123);
document.write(b);
</script>
</head>
<body bgcolor=”red” text=”blue”>
```

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

o/p-

7.slice():- used to returns selected elements in an array as new array elements.

Ex:-

```
<html>
<head>
<title>array</title>
<script language=”JavaScript”>
var a=new Array(9,33,12,90,6);
var b=a.slice(1,3);
document.write(b);
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

o/p-

8.shift():- used to delete first element from the array.

Ex:-

```
<html>
<head>
<title>array</title>
<script>
var a=new Array(9,33,12,90,6);
var b=a.shift();
```

```
document.write(a);
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

o/p-

9.unshift():- used to add elements at the beginning of the array.

Ex:-

```
<html>
<head>
<title>array</title>
<script language=”JavaScript”>
var a=new Array(9,33,12,90,6);
var b=a.unshift(12,8);
document.write(a);
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

o/p-

10.concat():- used to concatenate/joint two different array to produce a single array.

Ex:-

```
<html>
<head>
```

```
<title>array</title>
<script language=”JavaScript”>
var a=new Array(9,33,12,90);
var b=new Array(25,1,123);
var c=a.concat(b);                    //var c=b.concat(a);
document.write(c);
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

```
o/p-9,33,12,90,25,1,123
```

5. Functions

What are functions:-

Functions are the self-contained block of codes which are used to perform specific operation over the data. Functions are used to divide complex programs into simple parts. There are two types of functions available in java-script, that are:

1. User-defined functions
2. Pre-defined functions

User-defined functions:-

The functions which are declared and defined by the user at the time of writing a program are called user-defined functions. User-defined functions contain different components like:

i. A keyword “function” to define a function.

ii. A user-defined function name followed by a pair of parenthesis.

iii. Number of arguments to be passed to the function for operations.

iv. Body of the function

v. A return statement to return a value to the function.

Syn:-

```
function function-name(arguments)
{
//statements;
return statement;
}
```

Write a program to add two number using function?

```
<html>
<head>
<title>array</title>
<script language=”JavaScript”>
function addition(a,b)
{
c=a+b;
return(c);
}
document.write(“addition value=”+addition(5,4));
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

Write a program to find the factorial of a number using function ?

```
<html>
<head>
```

```
<title>array</title>
<script language=”JavaScript”>
function factorial(n)
{
var f=1;
while(n!=0)
{
f=f*n;
n--;
}
return(f);
}
document.write(“factorial value=”+factorial(5));
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

Pre-defined functions:-

The functions/methods which are already available inside java-script library and carry some meaning for the java-script interpreter are called pre-defined functions. All the java-script pre-defined functions are accessed by the help of pre-defined object.

1. math functions
2. conversion functions
3. user input functions
4. date functions

Math Functions:-

All math functions are called by the help of pre-defined object “Math” object by the help of dot(.) operator.

1.**pow(m,e):-** used to calculate power value by providing mantissa and exponent.

Ex:- document.write(Math.pow(5,2)); **o/p-25**

2.**sqrt(value):-** used to calculate square root value of a number.

Ex:- document.write(Math.sqrt(576)); **o/p-24**

3.**ceil(value):-** used to calculate ceiling value by increasing to next higher round-off value.

Ex:- document.write(Math.ceil(10.1)); **o/p-11**

4.**floor(value):-** used to calculate flooring value by decreasing it to next nearest round-off value.

Ex:- document.write(Math.floor(10.9)); **o/p-10**

5.**round(value):-** used to round off a number according to round-off rule.

Ex:- document.write(Math.round(5.7)); **o/p-6**

6.**abs(value):-** used to display absolute value of a number.

Ex:- document.write(Math.abs(-15)); **o/p-15**

7.**sin(radian):-** used to display sin theta value.

Ex:- document.write(Math.sin(0)); **o/p-0**

8.**cos(radian):-** used to display cos theta value.

Ex:- document.write(Math.cos(0)); **o/p-1**

9.**tan(radian):-** used to display tan theta value.

Ex:- document.write(Math.tan(45)); **o/p-1**

10.**max(value1,...,valuen):-** used to return maximum value from the list.

Ex:- document.write(Math.max(5,2,34,1,78)); **o/p-78**

11.**min(value1,…,valuen):-** used to return minimum value from the list.

Ex:- document.write(Math.min(12,5,172,0,29)); **o/p-0**

12.**log(value,base):-** used to return logarithmic value from log table.

Ex:- document.write(Math.log(10,2)); **o/p-1**

Conversion Function:-

Use to convert one type of data into another type.

1.**parseInt():-** used to convert string type data into integer type.

Syn:- var <variable-name>=parseInt(variable-name);

2.**parseFloat():-** used to convert string value into float value.

Syn:- var <variable-name>=parseFloat(variable-name);

3.**eval():-** used to convert string value into integer/float values.

Syn:- var <variable-name>=eval(variable-name);

4.**toString():-** used to convert integer/float value into string values.

Syn:- var <variable-name>=toString(variable-name);

User-input Function:-

1. prompt():- used to take user-input from keyboard by creating boxes using this method.

Syn:- var <variable-name>=prompt(“interactive message”);

Wap to add two integer number using prompt method?

```
<html>
<head>
<title>array</title>
<script language=”JavaScript”>
var a=prompt(“enter first number”);
var b=parseInt(a);
var c=prompt(“enter second number”);
var d=parseInt(c);
var e=b+d;
document.write(“<h1>”+“addition value=”+e+”</h1>”);
</script>
</head>
```

```
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

2. confirm():-used to display a confirmation box before printing any message/output. This box contains two buttons; one is “ok” and another is “cancel”.

Syn:- confirm(“interactive message”);

Wap to add two integer numbers using prompt and confirm method?

```
<html>
<head>
<title>array</title>
<script language=”JavaScript”>
var a=prompt(“enter first number”);
var b=parseInt(a);
var c=prompt(“enter second number”);
var d=parseInt(c);
var e=b+d;
confirm(“would you like to print the output”);
document.write(“<h1>”+“addition value=”+e+”</h1>”);
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

3.alert():-used to display an alert box if the condition is false or anything goes wrong according to user’s requirement.

Syn:- alert(“interactive message”);

Wap for alert method?

```
<html>
<head>
<title>array</title>
<script language=”JavaScript”>
var a=prompt(“enter your age”);
var b=parseInt(a);
document.write(“<h1>”+“addition value=”+e+”</h1>”);
if(b>=18)
document.write(“<h1>”+“you are eligible for vote”+”</h1>”);
else
alert(“you are not eligible”);
</script>
</head>
<body bgcolor=”red” text=”blue”>
</body>
</html>
```

Timer Function:-

1.getDay():- used to catch system day name.

2. getDate():- used to catch system date.

3.getMonth():- used to catch system month.

4.getFullYear():- used to catch system year.

5.getHours():- used to catch system hours from system date.

6.getMinutes():- used to catch system minutes from system date.

7.getSeconds():- used to catch system seconds from system date.

8.getMilliseconds():- used to catch system milliseconds from system date.

9.getTime():- used to catch system time from system date.

Wap to display a digital clock using date functions ?

```
<html>
<head>
<title>array</title>
<script language=”JavaScript”>
function clock()
{
var d=new Date();
var h=d.getHours();
var m=d.getMinutes();
var s=d.getSeconds();
m=checkTime(m);
s=checkTime(s);
document.getElementById(‘txt’).innerHTML=h+”:”+m+”:”+s;
var t=setTimeout(clock,500);
}
function checkTime(i)
{
if(i<10)
{
i=”0”+i;
}
return(i);
}
</script>
</head>
```

```
<body onload=”clock()” >
<div id=”txt”></div>
</body>
</html>
```

6. Validation:-

It is the type of restriction where user/programmer wants to provide some authentication/authorization to the page. It can be provided in two different ways:

1. empty validation:-

Wap to validate a login page ?

```
<html>
<head>
<title>login page validation</title>
<script language=”JavaScript”>
function myvalidate()
{
var id=document.form1.text1.value;
var pwd=document.form1.text2.value;
if(id==null||pwd==null)
alert(“login failed”);
else
document.write(“<h1>”+“your id=”+id+”</h1>”);
document.write(“<h1>”+“your pwd=”+pwd+”</h1>”);
}
</script>
</head>
<body bgcolor=”red” text=”blue”>
```

```
<font size=5 face=”lucida handwriting”><marquee bgcolor=”grey”>welcome to login page</marquee></font>
<h1 align=”center”><u>login info</u></h1>
<center>
<form name=”form1” method=”POST” action=””>
<table height=150 width=250>
<tr>
<td>Enter id:</td><td><input type=”text” name=”text1” placeholder=”enter your id”></td>
</tr>
<tr>
<td>Enter pwd:</td><td><input type=”password” name=”text2” placeholder=”enter your password”></td>
</tr>
<tr>
<td align=”center”><input type=”button” value=”submit” onclick=”myvalidate()”></td><td align=”center”><input type=”button” value=”reset”></td>
</tr>
<tr>
<td><input type=”checkbox”>stay sign-in</td><td align=”right”><a href=””>forgotten password ?</a></td>
</tr>
</table>
</form>
</center>
</body>
</html>
```

2. inline validation:-

Wap to validate a login page ?

```
<html>
```

```
<head>
<title>login page validation</title>
<script language=”JavaScript”>
function myvalidate()
{
var id=document.form1.text1.value;
var pwd=document.form1.text2.value;
if(id==null||pwd==null)
alert(“login failed”);
else
document.write(“<h1>”+“your id=”+id+”</h1>”);
document.write(“<h1>”+“your pwd=”+pwd+”</h1>”);
}
</script>
</head>
<body bgcolor=”red” text=”blue”>
<font size=5 face=”lucida handwriting”><marquee bgcolor=”grey”>welcome to login page</marquee></font>
<h1 align=”center”><u>login info</u></h1>
<center><form name=”form1” method=”POST” action=””>
<table height=150 width=250>
<tr>
<td>Enter id:</td><td><input type=”text” id=”text1” placeholder=”enter your id”></td>
</tr>
<tr>
<td>Enter pwd:</td><td><input type=”password” id=”text2” placeholder=”enter your password”></td>
</tr>
```

```
<tr>
<td align=”center”><input type=”button” value=”submit” onclick=”myvalidate()”></td><td align=”center”><input type=”button” value=”reset”></td>
</tr>
<tr>
<td><input type=”checkbox”>stay sign-in</td><td align=”right”><a href=””>forgotten password ?</a></td>
</tr>
</table>
</form>
</body>
</html>
```

Wap to add two number using textbox ?

```
<html>
<head>
<title>abc</title>
<script>
function add()
{
var a=parseInt(document.getElementById("t1").value);
var b=parseInt(document.getElementById("t2").value);
var c=document.getElementById("t3");
c.value=a+b;
}
</script>
</head>
<body>
```

```
Enter first no.:<input type="text" id="t1" name="t">
Enter second no.:<input type="text" id="t2" name="t2">
<input type="submit" name="submit" value="add" onclick="add()">
Addition value:<input type="text" id="t3" name="t3">
</body>
</html>
<html>
<head>
<title>abc</title>
<script>
function add()
{
var a=parseInt(document.f1.t1.value);
var b=parseInt(document.f1.t2.value);
var c=a+b;
document.write(c);
}
</script>
</head>
<body>
<form name="f1" method="post">
Ente first no.:<input type="text" name="t1">
Enter second no.:<input type="text" name="t2">
<input type="submit" name="submit" value="add" onclick="add()">
</body>
</html>
```

7. Objects and Events

Objects:-

1. Window:- it consists of following methods and properties.

Methods:-

prompt(),confirm(),alert(),setInterval(),clearInterval()

Properties:-

closed,document,history,location,

2. Document:-it consists of following methods and properties.

Methods:-

write()

Properties:-

linkColor,alinkColor,vlinkColor,bgColor,fgColor,Form,Image[],lastModified

3. History:-it consists of following methods and properties.

Methods:-

back(),forward(),go()

Properties:-

length

4. Form:-it consists of following methods and properties.

Methods:-

submit(),reset()

Properties:-

elements[]

5. Image:-it consists of following methods and properties.

Methods:-

Image()

Properties:-

src,height,width,border,name

6. Array:-it consists of following methods and properties.

Methods:-

sort(),reverse,join,splice(),push(),pop(),shift(),unshift()

Properties:-

length

7. Date:-it consists of following methods and properties.

Methods:-

getDay(),getDate(),getMonth(),getYear(),getMinutes(),getSeconds(),getMilliseconds(),getTime()

8. Elements[]:-it consists of following methods and properties.

Properties:-

text,password,button,radio,checkbox,textarea,submit,reset

9. Options[]:-it consists of following methods and properties.

Properties:-

Index,text,value

10. Math:-it consists of following methods and properties.

Methods:-

pow(),sqrt(),ceil(),floor(),abs(),round(),sin(),cos(),tan(),log(),max(),min()

Properties:-

E,PI

11. String:-it consists of following methods and properties.

Methods:-

substring(),lastIndexOf(),indexOf(),charAt(),charCodeAt(),toLowerCase(),toUpperCase()

12.Global:-it consists of following methods and properties.

Methods:-

parseInt(),parseFloat()

Events:-

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page. 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.

1. onabort:-

2. onblur:-

```
<html>
<head>
<title>abc</title>
<script>
function abc()
{
var x=document.getElementById("txt");
x.value=x.value.toUpperCase();
}
</script>
<body>
Enter a character:<input type="text" id="txt" onblur="abc()">
</body>
</html>
```

3. onchange:-

```
<html>
<head>
<title>abc</title>
<script>
```

```
function xyz()
{
var x=document.getElementById("combo").value;
document.getElementById("dropdown").innerHTML = "You selected:"+x;
}
</script>
<body>
<select id="combo" onchange="xyz()">
<option value="January">January
<option value="February">February
<option value="March">March
</select>
<p id="dropdown"></p>
</body>
</html>
```

4. onclick:-

```
<html>
<head>
<title>abc</title>
<script>
function abc()
{
window.open(“abc.html”);
}
</script>
</head>
```

```
<body>
<input type=”button” value=”click me” onclick=”abc()”>
</body>
</html>
```

5. ondblclcik:-

```
<html>
<head>
<title>abc</title>
<script>
function abc()
{
Document.bgColor=”red”;
}
function def()
{
Document.bgColor=”green”;
}
function ghi()
{
Document.bgColor=”blue”;
}
</script>
</head>
<body>
<input type=”button”  value=”btn1”  ondblclick=”abc()”>
<input type=”button”  value=”btn2”  ondblclick=”def()”>
```

```
<input type=”button”  value=”btn3”  ondblclick=”ghi()”>
</body>
</html>
```

6. onerror:-

```
<html>
<head>
<title>abc</title>
<script>
function abc()
{
alert('The image could not be loaded.');
}
</script>
</head>
<body bgcolor=”red” text=”blue”>
<img src="image.gif" onerror="abc()">
</body>
</html>
```

7. onfocus:-

```
<html>
<head>
<title>abc</title>
<script>
function abc(x)
{
x.style.background=”red”;
}
</script>
<body>
Enter a character:<input type="text" id="txt" onfocus="abc(this)">
</body>
</html>
```

8. onkeydown:-

```
<html>
<head>
<title>abc</title>
<script>
function abc()
{
alert(“a key is pressed”);
}
</script>
<body>
Enter a character:<input type="text" id="txt" onkeydown="abc()">
</body>
</html>
```

9. onkeypress:-

```
<html>
<head>
<title>abc</title>
<script>
function abc()
```

```
{
alert(“a key is pressed”);
}
</script>
<body>
Enter a character:<input type="text" id="txt" onkeypress="abc()">
</body>
</html>
```

10. onkeyup:-

```
<html>
<head>
<title>abc</title>
<script>
function abc()
{
var x=document.getElementById("txt");
x.value=x.value.toUpperCase();
}
</script>
<body>
Enter a character:<input type="text" id="txt" onkeyup="abc()">
</body>
</html>
```

11. onload:-

Example done in timer function.

12. onmousedown:-

13. onmousemove:-

14. onmouseout:-

15. onmouseover:-

```
<html>
<head>
<title>abc</title>
<script>
function over()
{
document.bgColor=”red”;
}
function out()
{
document.bgColor=”green”;
}
</script>
</head>
<body>
<p onmouseover=”over()” onmouseout=”out()”>
Hello students !<br>
It is javascript class.<br>
We are doing event handling
</p>
</body>
</html>
```

16. onmouseup:-

17. onselect:-

```
<html>
<head>
<title>abc</title>
<script>
function abc(x)
{
alert(“you have selected some text”);
}
</script>
<body>
Enter a character:<input type="text" id="txt" onselect="abc()">
</body>
</html>
```

18. onsubmit:-

```
<html>
<head>
<title>abc</title>
<script>
function abc()
{
alert(“the form was submitted”);
}
</script>
<body>
Enter a character:<input type="text" id="txt" onfocus="abc(this)">
```

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

19. onunload:-

20. ondrag:-

21. ondrop:-

22. oninvalid:-

23. onmessage:-

24. onprogress:-

25. onredo:-

26. onscroll:-

27. onundo:-

28. onwaiting:-

29. oninput:-

```
<html>
<head>
<title>abc</title>
<script>
function myFunction()
{
var x=document.getElementById("txt1").value;
document.getElementById("demo").innerHTML = "You wrote: " + x;
}
</script>
</head>
<body>
<input type="text" id="txt1" oninput="myFunction()">
```

```
<p id="demo"></p>
</body>
</html>
```

30. onplay:-

31. onpause:-

32. onplaying:-

33. onreset:-

```
<html>
<head>
<title>abc</title>
<script>
function abc()
{
alert(“the form was reset”);
}
</script>
</head>
<body>
<form onreset=”abc()”>
Enter your name:<input type=”text”>
<input type=”reset”>
</form>
</body>
</html>
```

DOM:-

Stand for Document Object Model. It is a standard for how to get, change, add, or delete HTML elements. It defines HTML elements as objects. It defines properties of all HTML elements. It defines methods to access all HTML elements. It defines events for all HTML elements.

DOM Methods:-

HTML DOM methods are actions you can perform (on HTML Elements).HTML DOM properties are values (of HTML Elements) that you can set or change. The HTML DOM can be accessed with JavaScript (and with other programming languages).In the DOM, all HTML elements are defined as objects. The programming interface is the properties and methods of each object. A property is a value that you can get or set (like changing the content of an HTML element).A method is an action you can do (like add or deleting an HTML element).

DOM Elements:-

1. **elements by id:-**

```
var myElement = document.getElementById("txt1");
```

2.Finding HTML elements by tag name:-

```
var x = document.getElementsByTagName("p");
```

3.Finding HTML elements by class name:-

```
var x = document.getElementsByClassName("intro");
```

4.Finding HTML elements by CSS selectors:-

```
var x = document.querySelectorAll("p.intro");
```

1. **Finding HTML elements by HTML object collections:-**

```
var x = document.forms["frm1"];
var text = "";
var i;
for (i = 0; i < x.length; i++)
{
text += x.elements[i].value + "<br>";
}
document.getElementById("txt1").innerHTML = text;
```

DOM Events:-

- When a user clicks the mouse→onclick event
- When a web page has loaded→onload event
- When an image has been loaded→onload event
- When the mouse moves over an element→onmouseover event
- When an input field is changed→onchange event
- When an HTML form is submitted→onsubmit event
- When a user strokes a key→keytyped

Ex:-

```
<html>
<head>
<title>abc</title>
</head>
<body>
<h1 onclick="this.innerHTML='Ooops!'">Click on this text!</h1>
</body>
</html>
```

Ex:-

```
<html>
<head>
<title>abc</title>
<script>
function mOver(obj)
{
obj.innerHTML = "Thank You";
}
function mOut(obj)
```

```
{
obj.innerHTML = "Mouse Over Me";
}
</script>
</head>
<body>
<div onmouseover="mOver(this)" onmouseout="mOut(this)"
style="background:red;width:120px;height:20px;padding:40px;">
Mouse Over Me</div>
</body>
</html>
```

Ex:-

```
<html>
<head>
<title>abc</title>
<script>
function mDown(obj)
{
obj.style.backgroundColor = "blue";
obj.innerHTML = "Release Me";
}
function mUp(obj)
{
obj.style.backgroundColor="red";
obj.innerHTML="Thank You";
}
```

```
</script>
</head>
<body>
<div onmousedown="mDown(this)" onmouseup="mUp(this)"
style="background:green;width:90px;height:20px;padding:40px;">
Click Me</div>
</body>
</html>
```