Full Stack Development Using Javascript-1

Unit-8 Javascript

8.1 Basics of Javascript: Client Side Scripting with JS, Overview, Characteristics and Advantages, Internal and External Javascript

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

JavaScript was first known as **LiveScript,** but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java. JavaScript made its first appearance in Netscape 2.0 in 1995 with the name **LiveScript**. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

Advantages of JavaScript:

The advantages of using JavaScript are the following:

- **Less server interaction** − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
- **Immediate feedback to the visitors** − They don't have to wait for a page reload to see if they have forgotten to enter something.
- **Increased interactivity** − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- **Richer interfaces** − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

Limitations of JavaScript:

We cannot treat JavaScript as a full-fledged programming language. It lacks the following important features:

- Client-side JavaScript does not allow the reading or writing of files. It has been kept for the security reason.
- JavaScript could not used for networking applications because there is no such support available.
- JavaScript doesn't have any multithreading or multiprocessor capabilities.

Syntax of JavaScript:

JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page.

You can place the **<script>** tags, containing your JavaScript, anywhere within your web page, but it is normally recommended that you should keep it within the **<head>** tags.

The <script> tag alerts the browser program to start interpreting all the text between these tags as a script. A simple syntax of your JavaScript will appear as follows.

The script tag takes two important attributes −

- **Language** − This attribute specifies what scripting language you are using. Typically, its value will be javascript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.
- **Type** − This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".

```
<script language = "javascript" type = "text/javascript">
   JavaScript code
</script>
```


8.2 Variables

JavaScript is an untyped language because in JavaScript the variables can hold any data type meaning that JavaScript does not have a type declaration and when the variable is created we do not need to specify any data type unlike other programming languages like Java, C#, C++, etc.

4 Ways to Declare a JavaScript Variable:

- Using var
- Using let
- Using const
- Using nothing

Example

```
<html>

<body>

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

var x = 16 + 4 +"xyz";

var y = "xyz" + 16 + 4;

document.write(x+"<br>");

document.write(y);

</script>
```

</body>

</html>

Output
20xyz
xyz164

Datatype

Primitive data types: The predefined data types provided by JavaScript language are known as primitive data types. Primitive data types are also known as in-built data types.

1. **String**
2. **Number**
4. **Boolean**
5. **Undefined**
6. **Null**

Non-primitive data types: The data types that are derived from primitive data types of the JavaScript language are known as non-primitive data types. It is also known as derived data types or reference data types.

1. **An object**
2. **An array**

Example

```
<html>
<body>
<script type="text/javascript" language="javascript">
var num=12;
var str="xyz";
var bl=true;
document.write(typeof(num));
document.write("<br>"+typeof(str));
document.write("<br>"+typeof(bl));
</script>
</body>
</html>
```

Output
number
string
boolean

Conditions: If, If…Else

If

Syntax

```
if(condition)
{
  // block of code to be executed if the condition is true
}
```

Example

```
<html>
<body>
<script type="text/javascript" language="javascript">
var age= 20;
if(age>=18)
{
  document.write("You are eligible for voting");
}
else
{
  document.write("You are not eligible for voting");
}
</script>
</body>
</html>
```


Output

You are eligible for voting

If…Else

Syntax

```
if(condition1)
{
  // block of code to be executed if condition1 is true
}
else if(condition2)
{
  // block of code to be executed if the condition1 is false and
condition2 is true
}
```

```
else
{
  // block of code to be executed if the condition1 is false and
condition2 is false
}
```

Example

```
<html>
<body>
<script type="text/javascript" language="javascript">
var sub="FSD-1";
if(sub=="FSD-1")
{
  document.write("ZPB");
}
else if(sub=="FCSP-1")
{
  document.write("VHA");
}
else if(sub=="ETC")
{
  document.write("JMA");
}
else if(sub=="PS")
{
  document.write("HRJ");
}
else
{
  document.write("Wrong Choice!!");
}
</script>
</body>
</html>
```

Output

```
ZPB
```

Loops: for, while, do…while

Types of Loops

1. **Entry Controlled –** while, for
2. **Exit Controlled –** do… while

While Loop

```
while(expression)
{
  Statement(s) to be executed if expression is true
}
```

Example

```
<html>
<body>
<script type="text/javascript" language="javascript">
        var count = 0;
        while (count < 10)
        {
          document.write("Current Count : " + count + "<br>");
          count++;
        }
</script>
</body>
</html>
```


Output

```
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Current Count : 5
Current Count : 6
Current Count : 7
Current Count : 8
Current Count : 9
```

For Loop

```
for (initialization; test condition; iteration statement)
{
  Statement(s) to be executed if test condition is true
}
```

Example

```
<html>
<body>
<script type="text/javascript" language="javascript">
        for(count = 0; count < 10; count++)
        {
              document.write("Current Count : " + count );
              document.write("<br >");
        }
</script>
</body>
</html>
```

Output

```
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Current Count : 5
Current Count : 6
Current Count : 7
Current Count : 8
Current Count : 9
```


Do..while Loop

```
do
{
  Statement(s) to be executed if test condition is true
}
while(expression);
```

Example

```
<html>
<body>
<script type="text/javascript" language="javascript">
          var count = 0;
```

```
      do
      {
            document.write("Current Count : " + count );
            document.write("<br>");
            count++;
      }
      while(count<10);
</script>
</body>
</html>
```

Output

```
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Current Count : 5
Current Count : 6
Current Count : 7
Current Count : 8
Current Count : 9
```


Break statement

Example

```
<html>
<body>
<script type="text/javascript" language="javascript">
        var x = 0;
       while (x < 10)
       {
         if (x == 5)
         {
           break;
         }
         x = x + 1;
         document.write( x + "<br>");
       }
```

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

Output

```
1
2
3
4
5
```

Continue statement

Example

```
<html>
<body>
<script type="text/javascript" language="javascript">
                    var x = 0;
          while (x < 10)
          {
            x = x + 1;
            if (x == 5)
            {
              continue;
            }
            document.write( x + "<br />");
          }
</script>
</body>
</html>
```


Output

```
1
2
3
4
6
7
8
9
10
```

Switch Case

Syntax

```
switch (expression)
{
  case condition 1: statement(s)
  break;

  case condition 2: statement(s)
  break;
  ...

  case condition n: statement(s)
  break;

  default: statement(s)
}
```

Example

```
<html>
<body>
<script type="text/javascript" language="javascript">
                    var grade = 'A';
        switch (grade)
        {
          case 'A': document.write("Good job<br />");
          break;

          case 'B': document.write("Pretty good<br />");
          break;

          case 'C': document.write("Passed<br />");
          break;

          case 'D': document.write("Not so good<br />");
          break;

          case 'F': document.write("Failed<br />");
          break;
```

```
        default:  document.write("Unknown grade<br />");
      }
</script>
</body>
</html>
```

Output

Good job

Functions: Syntax, Calling function on some event

A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again.

It helps programmers in writing modular codes. Functions allow a programmer to divide a big program into a number of small and manageable functions.

Before we use a function, we need to define it. The most common way to define a function in JavaScript is by using the **function** keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

Syntax

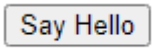
```
function functionname(parameter-list)
{
    Statements
}
```

Example (without parameter)

```
<html>
<head>
<script type="text/javascript" language="javascript">
      function Hello()
     {
        document.write ("Hello there!");
     }
</script>
</head>
<body>
        <input type = "button" onclick = "Hello()" value = "Say Hello">
```

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

Output



After clicking

Hello there!

Example (with parameter)

```
<html>
<head>
<script type="text/javascript" language="javascript">
                function info(name, age)
    {
       document.write (name + " is " + age + " years old.");
    }
</script>
</head>
<body>
        <input type = "button" onclick = "info('xyz',18)" value = "info">
</body>
</html>
```

Output

xyz is 18 years old.

Example

```
<html>
<head>
<script type="text/javascript" language="javascript">
      function info(first, last)
     {
        var full;
```

```
    full = first + last;
    return full;
  }
  function hello()
  {
    var result;
    result = info('Zalak', ' Bhatt');
    document.write(result);
  }
</script>
</head>
<body>
    <input type = "button" onclick = "hello()" value = "info">
</body>
</html>
```

Output

Zalak Bhatt

Javascript Array

1. **Javascript Array Literal**

Syntax

```
var arrayname = [value1, value2… valueN];
```

Example

```
<html>
<body>
        <script type="text/javascript" language="javascript">
  var emp =["abc", "pqr", "xyz"];
  for(i=0; i<emp.length; i++)
  {
  document.write(emp[i]+"<br>");
  }
```

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

Output

```
abc
pqr
xyz
```

2. Javascript Array “new” keyword

Syntax

```
var arrayname = new Array();
```

Example

```
<html>
<body>
<script type="text/javascript" language="javascript">
    var i;
    var emp = new Array();
    emp[0] = "abc";
    emp[1] = "xyz";
    emp[2] = "pqr";
    for(i=0; i<emp.length; i++)
    {
    document.write(emp[i]+"<br>");
    }
</script>
</body>
</html>
```

Output

```
abc
xyz
pqr
```

3. **Javascript Array constructor**

Example

```
<html>
<body>
<script type="text/javascript" language="javascript">
    var i;
    var emp = new Array("abc", "xyz", "pqr");
    for(i=0; i<emp.length; i++)
    {
    document.write(emp[i]+"<br>");
    }
</script>
</body>
</html>
```

Output

abc
xyz
pqr

Javascript Object

- Inbuilt Objects
- User Defined Objects

Inbuilt functions: Math, String, Date

Math: abs(), ceil(), floor(), sqrt(), pow()

Syntax

Math.method(numeric value);

Example

```
<html>
<body>
        <script type="text/javascript" language="javascript">
         document.write("MATH FUNCTIONS");
        document.write("<br/>Square root : "+Math.sqrt(9));
```

```
document.write("<br/>Absolute : "+Math.abs(-1));
document.write("<br/>Ceil : "+Math.ceil(-0.01));
document.write("<br/>Floor : "+Math.floor(-1.1));
document.write("<br/>Power: "+Math.pow(3,2));
</script>
</body>
</html>
```

Output

MATH FUNCTIONS
Square root : 3
Absolute : 1
Ceil : 0
Floor : -2
Power: 9

String: charAt(), charCodeAt(), concat(), indexOf(), split(), substr(), substring(), toLowerCase(), toUpperCase()

Sr.No.	Method & Description
1	charAt() Returns the character at the specified index.
2	charCodeAt() Returns a number indicating the Unicode value of the character at the given index.
3	concat() Combines the text of two strings and returns a new string.
4	indexOf() Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found.
5	lastIndexOf() Returns the index within the calling String object of the last occurrence of the specified value, or -1 if not found.

6	replace() Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring.
7	split() Splits a String object into an array of strings by separating the string into substrings.
8	substr() Returns the characters in a string beginning at the specified location through the specified number of characters.
9	substring() Returns the characters in a string between two indexes into the string.
10	toLowerCase() Returns the calling string value converted to lower case.
11	toUpperCase() Returns the calling string value converted to uppercase.

Example

```
<html>
<body>
        <script type="text/javascript" language="javascript">
        document.write("<br/>STRING FUNCTIONS");
        x="hello";
        y=" World";
        x1="this is demo";
        y1="is";
        document.write("<br/>char at 0th position : "+x.charAt(0));
        document.write("<br/>Unicode value at 0th position: "+x.charCodeAt(0));
        document.write("<br/>concated string : "+x.concat(y));
        document.write("<br/>first index occ: "+x1.indexOf(y1));
```

```
document.write("<br/>last index occ: "+x1.lastIndexOf(y1));
document.write("<br/>replace: "+x.replace("hell",y));
document.write("<br/>split: "+x.split("l"));
document.write("<br/>split: "+x1.split(" "));
document.write("<br/>sub str: "+x1.substr(2,5));
document.write("<br/>sub string: "+x1.substring(2,6));
document.write("<br/>sub string: "+x.toUpperCase());
document.write("<br/>sub string: "+y.toLowerCase());
</script>
</body>
</html>
```

Output

```
STRING FUNCTIONS
char at 0th position : h
Unicode value at 0th position: 104
concated string : hello World
first index occ: 2
last index occ: 5
replace: Worldo
split: he,,o
split: this,is,demo
sub str: is is
sub string: is i
sub string: HELLO
sub string: world
```


Date Manipulation

Syntax

```
var d1 = new Date();
```

Example

```
<html>
<body>
    <script type="text/javascript" language="javascript">
    var d1 = new Date();
     document.write(d1);
    </script>
</body>
</html>
```

Output

```
Wed Feb 15 2023 08:51:46 GMT+0530 (India Standard Time)
```

Sr.No.	Method & Description
1	Date() Returns today's date and time
2	getDate() Returns the day of the month for the specified date according to local time.
3	getDay() Returns the day of the week for the specified date according to local time.
4	getFullYear() Returns the year of the specified date according to local time.
5	getHours() Returns the hour in the specified date according to local time.
6	getMilliseconds() Returns the milliseconds in the specified date according to local time.
7	getMinutes() Returns the minutes in the specified date according to local time.
8	getMonth() Returns the month in the specified date according to local time.
9	getSeconds() Returns the seconds in the specified date according to local time.
10	getTime() Returns the numeric value of the specified date as the number of milliseconds since January 1, 1970, 00:00:00 UTC.
11	setDate() Sets the day of the month for a specified date according to local time.

12	setFullYear() Sets the full year for a specified date according to local time.
13	setHours() Sets the hours for a specified date according to local time.
14	setMilliseconds() Sets the milliseconds for a specified date according to local time.
15	setMinutes() Sets the minutes for a specified date according to local time.
16	setMonth() Sets the month for a specified date according to local time.
17	setSeconds() Sets the seconds for a specified date according to local time.
18	setTime() Sets the Date object to the time represented by a number of milliseconds since January 1, 1970, 00:00:00 UTC.

Example

```
<html>
<body>
<script type="text/javascript" language="javascript">
    var d1 = new Date();
    document.write(d1+"<br>");
    document.write(d1.getDay()+"<br>");
    document.write(d1.getMonth()+"<br>");
    document.write(d1.getDate()+"<br>");
    document.write(d1.getFullYear()+"<br>");
    document.write(d1.getHours()+"<br>");
    document.write(d1.getMinutes()+"<br>");
```

```
    document.write(d1.getSeconds()+"<br>");
</script>
</body>
</html>
```

Output

```
Wed Feb 15 2023 08:59:07 GMT+0530 (India Standard Time)
3
1
15
2023
8
59
7
```

User Defined Objects:

1. **By Object Literal**

Syntax

```
object = {property1 : value1, property2 : value2,.. propertyN : valueN}
```

Example

```
<html>
<body>
        <script type="text/javascript" language="javascript">
        var emp ={id:101, name:"xyz", salary:60000};
        document.write(emp.id+" "+emp.name+" "+emp.salary);
        </script>
</body>
</html>
```

Output

```
101 xyz 60000
```

2. **By creating instance of object**

Synatx

```
var objectname = new Object();
```

Example

```
<html>
<body>
<script type="text/javascript" language="javascript">
    var emp = new Object();
    emp.id = 101;
    emp.name = "xyz";
    emp.salary = 60000;
    document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
</body>
</html>
```

Output

```
101 xyz 60000
```

3. **By using an object constructor**

Example

```
<html>
<body>
<script type="text/javascript" language="javascript">
    function emp(id, name, salary)
    {
    this.id = id;
    this.name = name;
    this.salary = salary;
    }
    e = new emp(101, "xyz", 60000);
    document.write(e.id+" "+e.name+" "+e.salary);
</script>
</body>
</html>
```

Output

```
101 xyz 60000
```

8.3 Pop up Boxes: Alert, Confirm, Prompt

Alert Box

An alert dialog box is mostly used to give a warning message to the users. For example, if one input field requires to enter some text but the user does not provide any input, then as a part of validation, you can use an alert box to give a warning message.

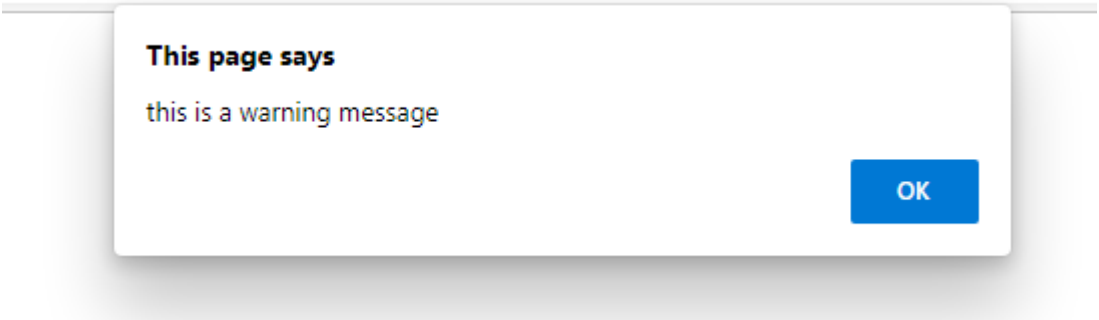
Nonetheless, an alert box can still be used for friendlier messages. Alert box gives only one button "OK" to select and proceed.

Example

```
<html>
<body>
        <script>
                function warn()
                {
                alert("this is a warning message");
                document.write("Warning!!!!");
                }
        </script>
        </head>

        <body>
                <input type="button" value="ok" onclick="warn()"/>
        </body>
</html>
```


Output



Confirmation Box

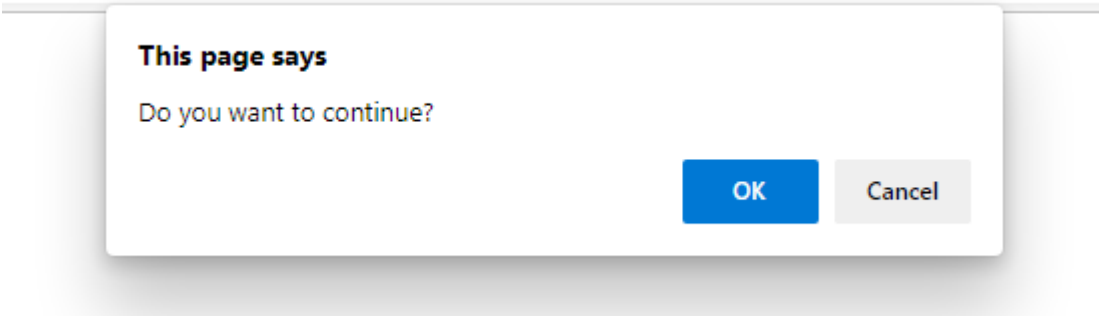
A confirmation dialog box is mostly used to take user's consent on any option. It displays a dialog box with two buttons: OK and Cancel.

If the user clicks on the OK button, the window method confirm() will return true. If the user clicks on the Cancel button, then confirm() returns false. You can use a confirmation dialog box as follows.

Example

```
<html>
	<head>
	<script type="text/javascript">
			function conf()
			{
				ret = confirm("Do you want to continue?");
				if(ret==true)
				{
					document.write("user wants");
					return true;
				}
				else
				{
					document.write("user does not want");
					return false;
				}
			}
	</script>
	</head>
	<body>
			<input type="button" onclick="conf()" value="OK"/>
	</body>
</html>
```

Output



Prompt Dialog Box

The prompt dialog box is very useful when you want to pop-up a text box to get user input. Thus, it enables you to interact with the user. The user needs to fill in the field and then click OK.

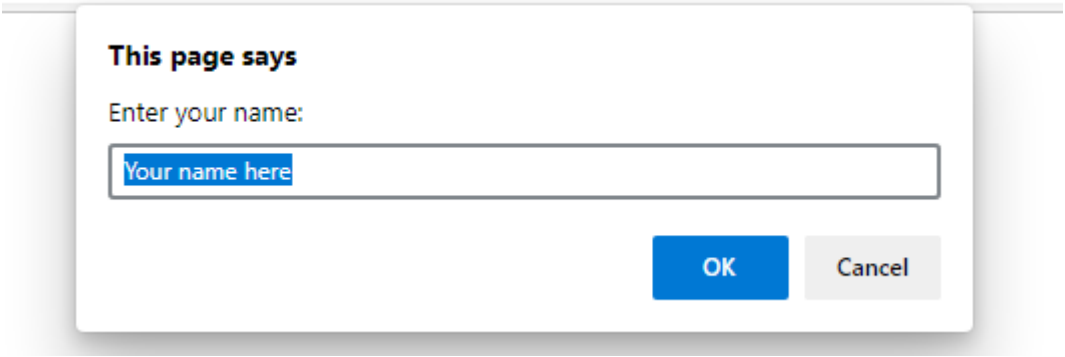
This dialog box is displayed using a method called prompt() which takes two parameters: (i) a label which you want to display in the text box and (ii) a default string to display in the text box.

This dialog box has two buttons: OK and Cancel. If the user clicks the OK button, the window method prompt() will return the entered value from the text box. If the user clicks the Cancel button, the window method prompt() returns null.

Example

```
<html>
	<head>
	<script type="text/javascript">
			function getval()
			{
				var ret = prompt("Enter your name: ", "Your name here");
				document.write("you entered "+ret);
			}
	</script>
	</head>
	<body>
	<input type="button" onclick="getval()" value="OK"/>
	</body>
</html>
```

Output



Example: Write a JS that find position of first occurrence of vowel ‘a’ and last occurrence of vowel ‘a’ in the given word “ajanta” also return the string between them

```
<html>
    <head>
    <script type="text/javascript">
        z="ajanta";
        x=z.indexOf("a");
        y=z.lastIndexOf("a");
        document.write("<br/>first index occ: "+x);
        document.write("<br/>last index occ: "+y);
        document.write("<br/>sub string: "+z.substring(x+1,y));
    </script>
    </head>
    <body>
    </body>
</html>
```

Output

first index occ: 0
last index occ: 5
sub string: jant

Example: Write a JavaScript that uses function to calculate how many days are left in your birthday.

```
<html>
    <head>
    <script type="text/javascript">
        function daysdifference(date1, date2)
        {
            one_day =24*60*60*1000;
            date1_ms= date1.getTime();
            date2_ms= date2.getTime();
            difference_ms = Math.abs(date1_ms-date2_ms);
            return(Math.round(difference_ms/one_day));
        }
    </script>
    </head>

    <body>
    <script>
        var d1 = new Date();
        d1.setDate(25);
        d1.setMonth(4);
        var d2 = new Date();
        document.write("<br/>No of days left :"+daysdifference(d1,d2));
        </script>
    </body>
</html>
```


Output

No of days left :98

Example: $1+ x/1! + x^2/2! + x^3/3!....$

```
<html>
<head>
<script type="text/javascript">
function fact(n)
{
if(n==1)
    return(1);
else
        {
                factorial=n*fact(n-1);
                return factorial;
        }
}
</script>
</head>
<body>
<script type="text/javascript">
x=3;
sum=0;
for(n=1;n<=4;n++)
{
        sum=sum+Math.pow(x,n)/fact(n);
}
total=1+sum;
document.write(total);
</script>
</body>
</html>
```

Example: Given digit is 23. divide it into two parts 2 and 3 and find 2^3

```
<html>
	<head>
	<script type="text/javascript">
		n=23;
		i=n%10;
		n=parseInt(n/10);
		document.write("Value of i:"+i);
		document.write("<br/>Value of n:"+n);
		document.write("<br/>ANSWER:"+Math.pow(n,i));
	</script>
	</head>
	<body>
	</body>
</html>
```

Output

```
Value of i:3
Value of n:2
ANSWER:8
```

Example: LUCKY NUMBER (hint : number = 7777 => 28 => 10 => 1) if final sum=1 then given number is lucky number

```
<html>
	<head>
	<script type="text/javascript">
			n=6661;
			while(n>9)
			{
				sum=0;
			while(n>0)
			{
			i=n%10;
			n=parseInt(n/10);
			sum=sum+i;
			}
			document.write(sum+"<br/>");
			n=sum;
			}
```

```
                    if(n==1)
                    {
                        document.write("NUMBER IS A LUCKY ONE");
                    }
                    else
                    {
                        document.write("NUMBER IS A UNLUCKY ONE");
                    }
        </script>
        </head>
        <body>
        </body>
</html>
```

Output

6661

19
10
1
NUMBER IS A LUCKY ONE