Java Script Objects

- A “**JavaScript**” object is an entity having **state and behavior** (properties and method).

For example: car, pen, bike, chair, glass, keyboard, monitor etc.

- It is an **object-based language.(**All OOP concepts except inheritance).
- JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

Types of Objects:

1. **Built-in Objects**
2. **User-Defined Objects**

Java Script Objects - Built-in Objects

1. JavaScript Native Objects/ Built-in Objects

- JavaScript Number Object
- JavaScript Boolean Object
- JavaScript String Object
- JavaScript Date Object
- JavaScript Math Object
- JavaScript Array Object

Java Script Objects – Number

- The JavaScript number object represent a **numeric value.**
- The **Number object** is a fundamental **wrapper object** that represents and manages number
- **Integers, decimal, or float point numbers,** among many other types of numbers, are all represented as number objects.
- Values of various kinds can be turned into numbers using the **Number()** method.

```
var n=new Number(value);
```

```
<html>
<body>
<script>
var x=102;//integer value
var y=102.7;//floating point value
var z=13e4;//exponent value, output: 130000
var n=new Number(16);//integer value by number
object
document.write(x+" "+y+" "+z+" "+n);
</script>
</body>
</html>
```

Java Script Objects – Number

Methods	Description
isFinite()	It determines whether the given value is a **finite number.**
isInteger()	It determines whether the **given value is an integer.**
parseFloat()	It converts the given string into a **floating point number.**
parseInt()	It converts the given string into an integer number.
toExponential()	It returns the string that represents exponential notation of the given number.
toFixed()	It returns the string that represents a number with exact digits after a decimal point.
toPrecision()	It returns the string representing a number of specified precision.
toString()	It returns the given number in the form of string.

Constant	Description
MIN_VALUE	returns the largest minimum value.
MAX_VALUE	returns the largest maximum value.
POSITIVE_INFINITY	returns positive infinity, overflow value.
NEGATIVE_INFINITY	returns negative infinity, overflow value.
NaN	represents "Not a Number" valu

Java Script Objects – String

- The **JavaScript string** is an object that represents a **sequence of characters.**

There are 2 ways to create string in JavaScript

1. **By string literal**
2. **By string object (using new keyword)**

1) By string literal

- The string literal is created **using double quotes**.

var str = "Hello"

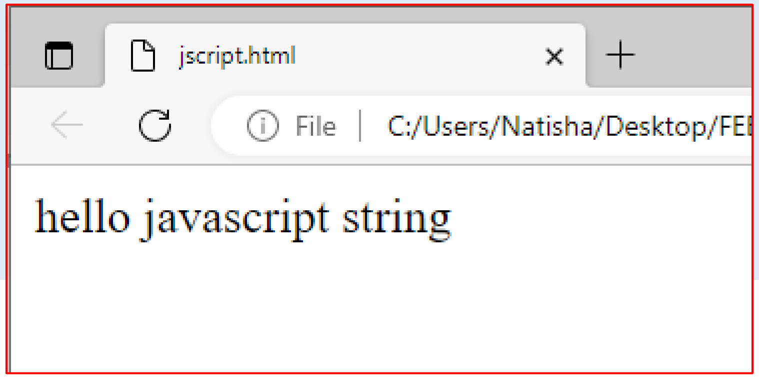
```
<html>
<body>
<script>
var str="This is string literal";
document.write(str);
</script>
</body>
</html>
```

2) By string object (using new keyword)

- The syntax of creating string object using new keyword is given below:

var str=new String("string literal");

```
<!DOCTYPE html>
<html>
<body>
<script>
var str=new String("hello javascript string");
document.write(str);
</script>
</body>
</html>
```



hello javascript string

JavaScript String Methods

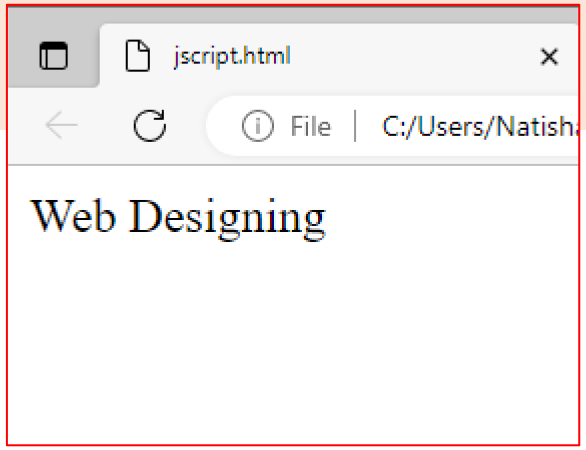
Methods	Description
charAt()	Return **char value present at the specified index**.
concat()	Return **combination of two or more strings.**
indexOf()	Return **position of a char value** present in the given string.
replace()	string with the **specified replacement.**
substr()	It is used to fetch the part of the given string on the basis of the specified starting position and length.
substring()	It is used to fetch the part of the given string on the basis of the specified index.
slice()	It is used to **fetch the part of the given string**
toLowerCase()	It converts the given **string into lowercase letter.**
toUpperCase()	It converts the given **string into uppercase letter.**
toString()	**Returns a string** representing the particular object.
split()	It **splits a string into substring array**, then returns that newly created array.
trim()	It **trims the white space from the left and right side** of the string.

String Methods

concat(str) method:

The JavaScript String concat(str) method **concatenates or joins two strings.**

```
<html>
<body>
<script>
var s1="Web ";
var s2="Designing";
var s3=s1.concat(s2);
document.write(s3);
</script>
</body>
</html>
```

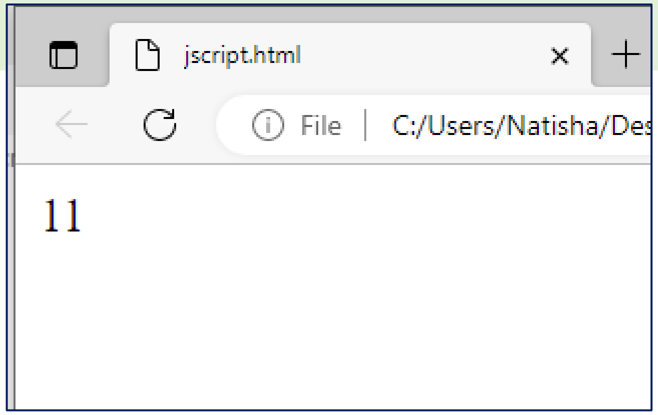


```
Web Designing
```

JavaScript String **indexOf(str)** Method

The JavaScript String indexOf(str) **method returns the index position of the given string**

```
<!DOCTYPE html>
<html>
<body>
<script>
var s1="javascript for web";
var n=s1.indexOf("for");
document.write(n);
</script>
</body>
</html>
```



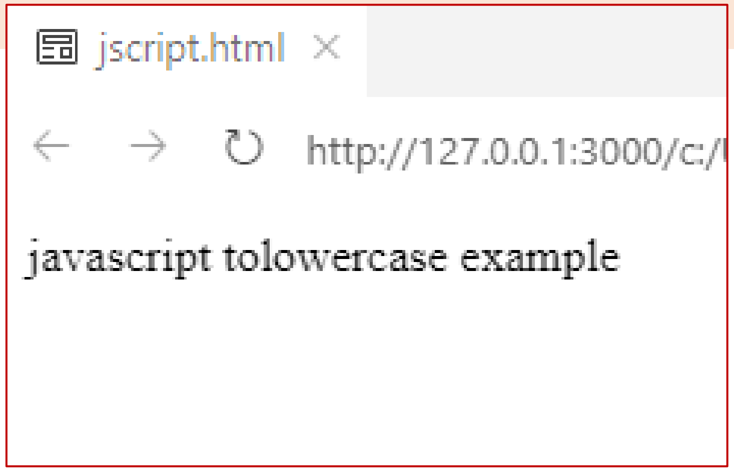
```
11
```

String Methods

JavaScript String toLowerCase() Method

The JavaScript String toLowerCase() method returns the given string in lowercase letters

```
<html>
<body>
<script>
var s1="JavaScript toLowerCase Example";
var s2=s1.toLowerCase();
document.write(s2);
</script>
</body>
</html>
```



jscript.html ×

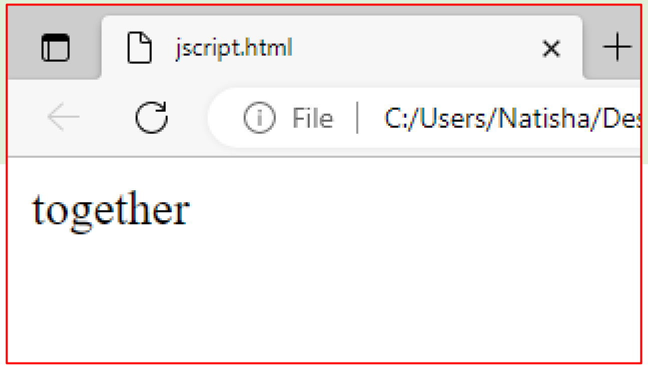
http://127.0.0.1:3000/c:/

```
javascript tolowercase example
```

7) JavaScript String slice(beginIndex, endIndex) Method

- **The JavaScript String slice(beginIndex, endIndex) method returns the parts of string from given beginIndex to endIndex.**
- **In slice() method, beginIndex is inclusive and endIndex is exclusive.**

```
<!DOCTYPE html>
<html>
<body>
<script>
var s1="We are together";
var s2=s1.slice(6,15);
document.write(s2);
</script>
</body>
</html>
```



jscript.html ×

File | C:/Users/Natisha/De

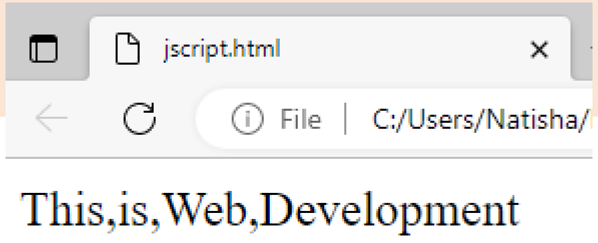
```
together
```

String Methods

JavaScript String split() Method

splits the given string.

```
<html>
<body>
<script>
var str="This is Web Development";
document.write(str.split(" "));
//ocument.write(s2);
</script>
</body>
</html
```

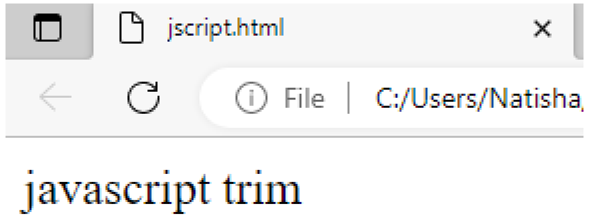


```
This,is,Web,Development
```

7) JavaScript String trim() Method:

trim() eliminates the spaces in the string

```
<html>
<body>
<script>
var s1="   javascript     trim    ";
var s2=s1.trim();
document.write(s2);
</script>
</body>
</html>
```



```
javascript trim
```

Java Script Boolean Object

Boolean

- JavaScript Boolean is an object that represents value in two states: **true or false.**
- You can create the JavaScript Boolean object by **Boolean()** constructor

```
Boolean b=new Boolean(value);
```

```
<script>
document.write(10<20);//true
document.write(10<5);//false
</script>
```

Java Script Objects - Date

Date:

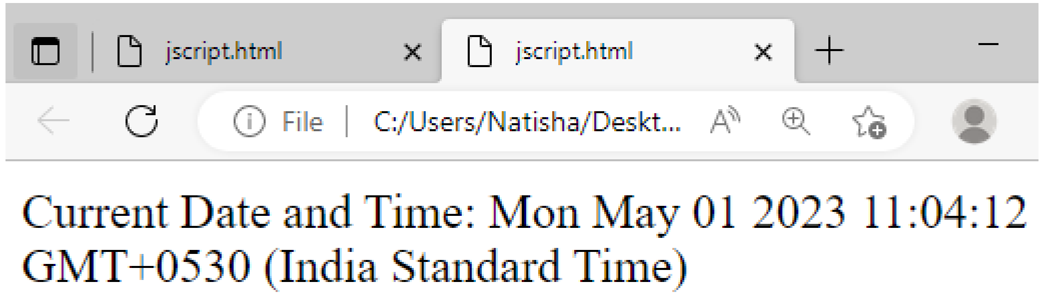
- The JavaScript date object can be used to **get year, month and day**.
- You can display a timer on the webpage by the help of JavaScript date object.

Constructor:

4 variant of Date constructor to create date object.

1. Date()
2. Date(milliseconds)
3. Date(dateString)
4. Date(year, month, day, hours, minutes, seconds, milliseconds)

```
<html>
<body>
Current Date and Time:
<span id="txt"></span>
<script>
var today=new Date();
document.getElementById('txt').innerHTML=today;

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

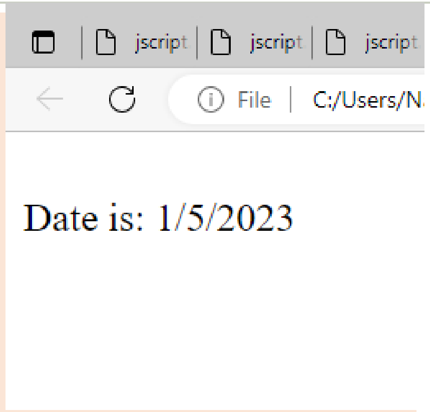


Current Date and Time: Mon May 01 2023 11:04:12 GMT+0530 (India Standard Time)

Java Script Objects -Date

Methods	Description
getDate()	It returns the integer value between **1 and 31**
getDay()	It returns the integer value between **0 and 6**
getFullYears()	**represents the year** on the basis of local time
getMonth()	It returns the integer value between **0 and 11** that represents the month

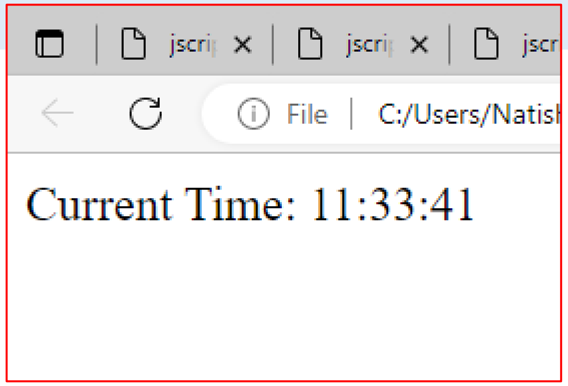
```
<!DOCTYPE html>
<html>
<script>
    var date=new Date();
    var day=date.getDate();
    var month=date.getMonth()+1;
    var year=date.getFullYear();
document.write("<br>Date is: "+day+"/"+ month+"/"+year);
    </script>
    </body>
</html>
```



Date is: 1/5/2023

JavaScript Current Time Example

```
<html>
<body>
Current Time: <span id="txt"></span>
<script>
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
document.getElementById('txt').innerHTML=h+":"+m+":"+s;
</script>
</body>
</html>
```



Current Time: 11:33:41

Java Script Objects – Math Object

- The JavaScript math object provides **several constants and methods to perform mathematical operation.**

```
let num1 = 16;
let sq = Math.sqrt(num1);

let num2 = 3.14;
let rounded = Math.round(num2);

// Generate a random number between 0 and 1
let random = Math.random();

// Calculate the maximum of two numbers
let num4 = 20;
let max = Math.max(num1, num4);

// Calculate the minimum of two numbers
let num5 = -5;
let min = Math.min(num2, num5);
```

```
The square root of 16 is 4

3.14 rounded to the nearest integer is 3



A random number between 0 and 1 is

0.9191471923452968



The maximum of 16 and 20 is 20



The minimum of 3.14 and -5 is -5
```

Java Script Objects

2. User-Defined Objects

- The new operator is used to create an instance of an object.
- To create an object, the new operator is followed by the constructor method

Syntax

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

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

- **The Object() Constructor**
  - A constructor is a function that **creates and initializes an object**.
  - JavaScript provides a special constructor function called **Object()** to build the object

Creating and assigning properties to the object

```
var emp=new Object();

emp.id=101;
emp.name="Naveen Kumar";
emp.salary=50000.50;

document.write("emp name is : " + emp.name + "<br>");
 document.write("salary is : " + emp.salay+ "<br>");
```

Java Script Objects

- create a user defined object using object literal notation

```
// creating an object using object literal notation

const person = {
    firstName: “naveen",
    lastName: “reddy",
    age: 30,
    occupation: "Developer"
  };

 // accessing object properties using dot notation
  console.log(person.firstName);
  console.log(person.age);
```

//const creates "constant" array that cannot be reassigned another

- create an object with a User-Defined object using a constructor

```
function book(title, author) //constructor
{
        this.title = title;
        this.author  = author;
}
```

```
//creating array as object using constructor
var myBook = new book("JAVA", “Naveen");
      document.write("Book title is : " + myBook.title );
    document.write("Book author is : " + myBook.author );
```

Java Script Objects- Defining Methods for an Object

Defining Methods in constructor

```
<html>
<head>
</head>
<body><h3>To add a method to a JavaScript
object.</h3>
<script>
function Calculator(){
//adding the another method
   Calculator.prototype.add = function (a,b)
       {
       var result = a+b;
       document. Writeln("sum is:"+result)
       }
 }
 var calc = new Calculator();
 calc.add(10,20);
    </script>
</body>
</html>
```

Adding the method as a property

```
<html><head></head>
<body>
<h3>To add a method to a JavaScript
object.</h3>
<p id="method-to-obj"></p>
<script>
   function Car(name, model, year, color) {
      this.Name = name;
      this.Model = model;
      this.Year = year;
      this.Color = color;
   }
    var car1 = new Car("Maruti", "Vitara Brezza", "2016", "Red");
  car1.prop = function() //added as property
   {
     document.writeln(""+this.Name+" has launched in"+this.Year);
   }
    car1.prop();
</script>
</body>
</html>
```

JavaScript Arrays

- JavaScript array is an object that represents a **collection of similar type of elements.**

3 ways to construct array in JavaScript

1. **By array literal**
2. **By creating instance of Array directly (using new keyword)**
3. **By using an Array constructor (using new keyword)**

1) JavaScript array literal:

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

```
Ex1.
var cars = ["Saaboo","Volvo","BMW"]; (or)
const cars = ["Saaboo","Volvo","BMW"];
```

```
Ex2.
var cars = [];
        cars[0]= "Saab";
        cars[1]= "Volvo";
        cars[2]= "BMW";
```

```
<!-- CREATING AN ARRAYA AND ACCESSING THE ELEMENTS-->
<html>
<body>
<script>
var emp=["Shivam","Vansh","Sameer"];
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br/>");
}
</script>
</body>
</html>
```

JavaScript Array directly (new keyword)

Creating an array using new keyword:

var arr=new Array();

```
<html>
<body>
<script>
var i;
var emp = new Array();
emp[0] = “kiran";
emp[1] = “Uday";
emp[2] = “Ram";

for (i=0;i<emp.length;i++)
{
        document.write(emp[i] + "<br>");
}
</script>
</body>
</html>
```

Creating an array using constructor

- create instance of array by passing arguments in constructor so that no need to provide value explicitly.

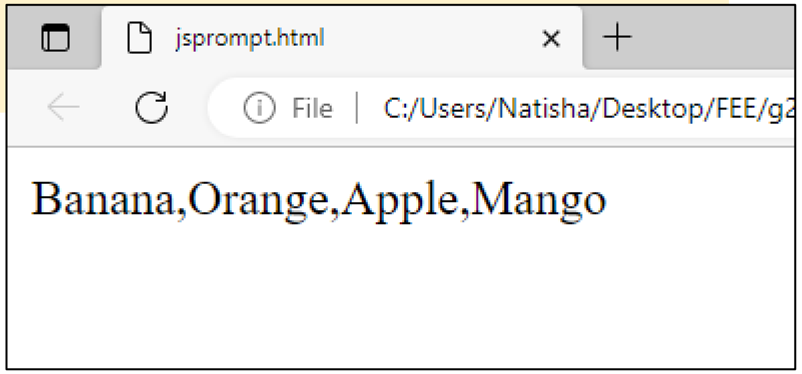
```
<html>
<body>
<script>
var emp=new Array(“vansh",“shiva",“sneha");
for (i=0;i<emp.length;i++)
{
        document.write(emp[i] + "<br>");
}
</script>
</body>
</html>
```

JavaScript Array Methods

toString() method

- **converts an array to a string** of (comma separated) array values.

```
<html>
<body>
<h2>JavaScript Array Methods</h2>
<h2>toString()</h2>
<p>The toString() method returns an array as a
comma separated string:</p>
<p id="demo"></p>
<script>
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.toString();
</script>
</body>
</html>
```



jsprompt.html

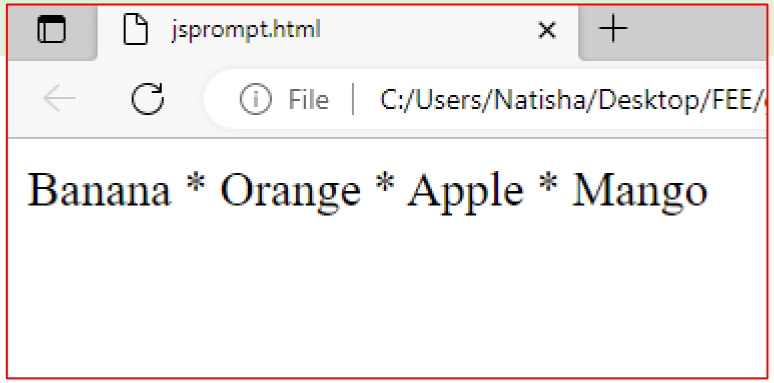
File | C:/Users/Natisha/Desktop/FEE/g2

```
Banana,Orange,Apple,Mango
```

Join():

- Joins all array elements into a string.
- Also, can **specify the separator**

```
<html>
<body>
<p id="demo"></p>
<script>
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.join(" * ");
</script>
</body>
</html>
```



jsprompt.html

File | C:/Users/Natisha/Desktop/FEE/(

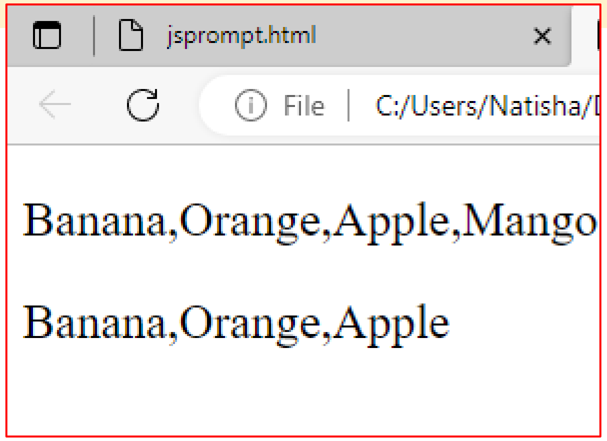
```
Banana * Orange * Apple * Mango
```

JavaScript Array Methods

pop()

- **removes the last element** from an array:

```
<html>
<body>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = fruits;
fruits.pop();
document.getElementById("demo2").innerHTML = fruits;
</script>
</body>
</html>
```



jsprompt.html

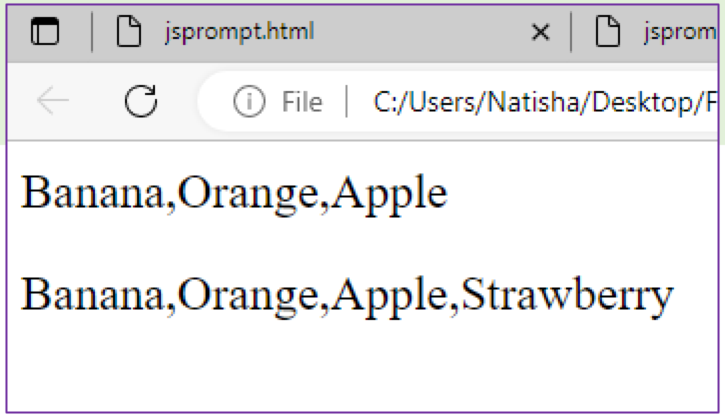
File | C:/Users/Natisha/

```
Banana,Orange,Apple,Mango

Banana,Orange,Apple
```

push():

- **adds a new element** to an array (at the end)

```
<html>
<body>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
const fruits = ["Banana", "Orange", "Apple"];
document.getElementById("demo1").innerHTML = fruits;
fruits.push("Strawberry");
document.getElementById("demo2").innerHTML = fruits;
</script>
</body>
</html>
```



jsprompt.html

File | C:/Users/Natisha/Desktop/F

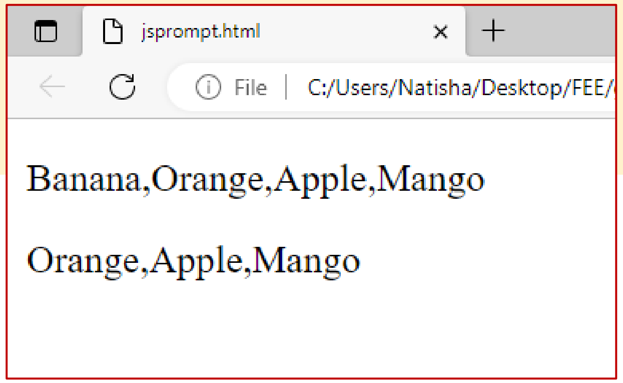
```
Banana,Orange,Apple

Banana,Orange,Apple,Strawberry
```

JavaScript Array Methods

shift()

- **removes the first array element** and "shifts" all other elements to a lower index

```
<html>
<body>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = fruits;
fruits.shift();
document.getElementById("demo2").innerHTML = fruits;
</script>
</body>
</html>
```

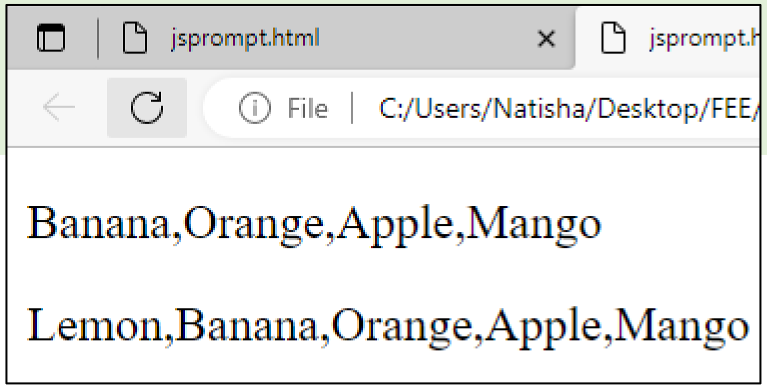


```
Banana,Orange,Apple,Mango

Orange,Apple,Mango
```

unshift():

- **adds a new element to an array (at the beginning),** and "unshifts" older elements:

```
<html>
<body>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
const fruits = ["Banana", "Orange", "Apple",
"Mango"];
document.getElementById("demo1").innerHTML = fruits;
fruits.unshift("Lemon");
document.getElementById("demo2").innerHTML = fruits;
</script>
</body>
</html>
```



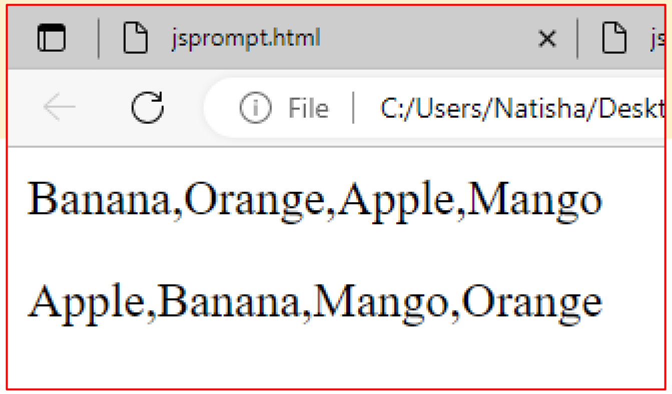
```
Banana,Orange,Apple,Mango

Lemon,Banana,Orange,Apple,Mango
```

JavaScript Array Methods

sort()

- sorts an array alphabetically

```
<html>
<body>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = fruits;
fruits.sort();
document.getElementById("demo2").innerHTML = fruits;
</script>
</body>
</html>
```

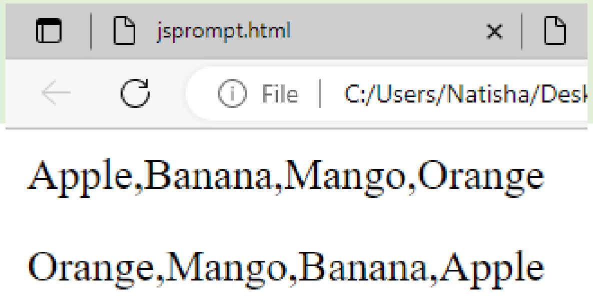


```
Banana,Orange,Apple,Mango

Apple,Banana,Mango,Orange
```

reverse():

- Reverse the elements of the array

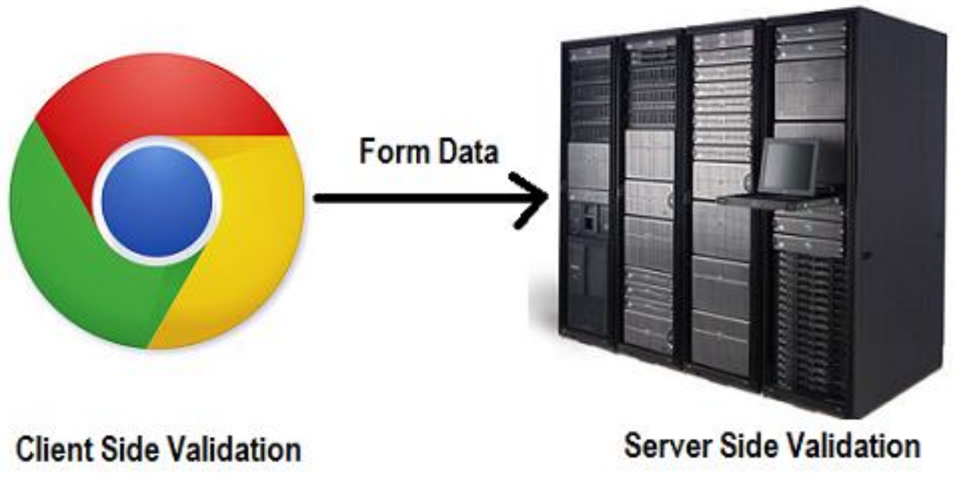
```
<html> <body>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
// Create and display an array:
const fruits = ["Banana", "Orange", "Apple",
"Mango"];
// First sort the array
fruits.sort();
document.getElementById("demo1").innerHTML = fruits;
fruits.reverse();
document.getElementById("demo2").innerHTML = fruits;
</script>
</body>
</html>
```



```
Apple,Banana,Mango,Orange

Orange,Mango,Banana,Apple
```

FORM VALIDATION

- **Validations can be performed on the server side or on the client side (web browser)**



- **Client side validation** is an **initial check** and an important feature of **good user experience**
- by catching and requiring corrections **to invalid data before** it is sent to the server to be rejected there,
- the delay caused by a round trip to the server for server-side validation is avoided.

- **Server-side validation** is necessary to check data sent to the server, **ensuring incorrect or malicious data is rejected.**

FORM VALIDATION

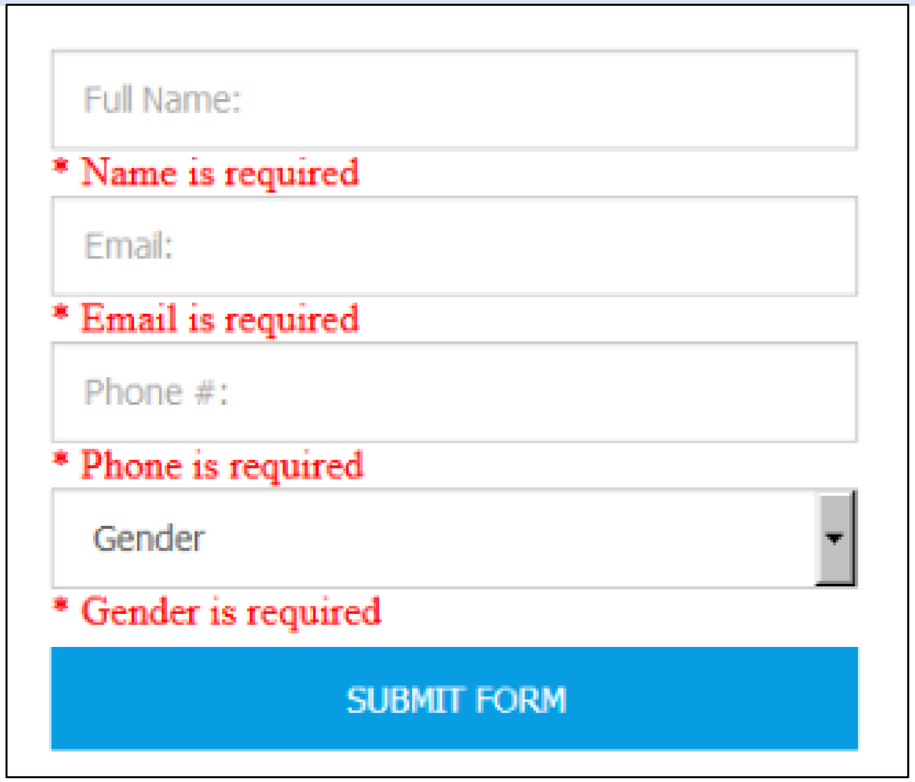
- JavaScript provides a way to **validate form's data on the client's computer** before sending it to the web server.
- Form validation generally performs **two functions**.

1) Basic Validation – First of all, the form must be checked to make sure **all the mandatory fields are filled in**.

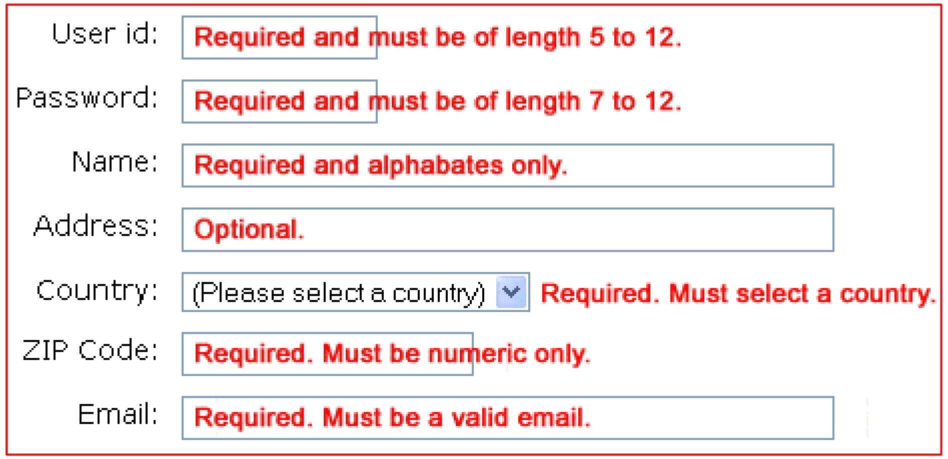
- It would require just a loop through each field in the form and check for data.

2) Data Format Validation – Secondly, the **data that is entered must be checked for correct** form and value.

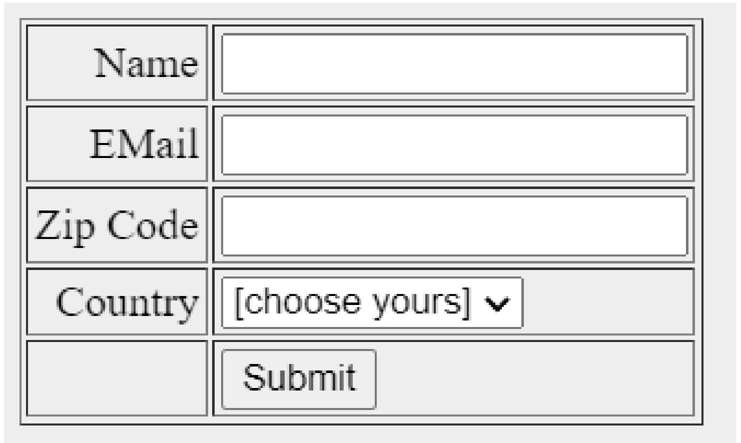
- Your code must include appropriate logic to test correctness of data.



Full Name:
* Name is required
Email:
* Email is required
Phone #:
* Phone is required
Gender
* Gender is required
SUBMIT FORM



User id:	Required and must be of length 5 to 12.
Password:	Required and must be of length 7 to 12.
Name:	Required and alphabates only.
Address:	Optional.
Country:	(Please select a country) Required. Must select a country.
ZIP Code:	Required. Must be numeric only.
Email:	Required. Must be a valid email.



Name	
EMail	
Zip Code	
Country	[choose yours]
	Submit

FORM VALIDATION- name and password validation

Validate the name and password fields

- The name can't be empty and password can't be less than 6 characters long.

```
<script>
 function validate() {
    var username =document.getElementById("username").value;
    var password =document.getElementById("password").value;
  var usernameError =document.getElementById("usernameError");
  var passwordError = document.getElementById("passwordError");
      usernameError.innerHTML = "";
      passwordError.innerHTML = "";
         if (username.length < 6) {
usernameError.innerHTML = "Username must be at least 6 chars";
        return false;
}  if(password.length < 8) {
 passwordError.innerHTML = "Password must be at least 8 chars";
           return false;
          }
        return true;
       }
</script>
```

```
<body>
  <form name="myform" action="page2.html" onsubmit="return
validateform()" >
   <label for="username">Username:</label>
       <input type="text" id="username" name="username">
       <span id="usernameError" class="error"></span><br>

       <label for="password">Password:</label>
       <input type="password" id="password" name="password">
       <span id="passwordError" class="error"></span><br>

        <input type="submit" value="Submit">
     </form>
```

FORM VALIDATION- number validation

number validation

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Number Validation Example</h1>
    <form onsubmit="return validate()">
      <label for="number">Number:</label>
     <input type="text" id="number" name="number">
    <span id="numberError“ class="error"></span><br>
      <input type="submit" value="Submit">
    </form>
```

isNaN() tests if the number is the value NaN

```
<script>
function validate() {
 var number = document.getElementById("number").value;
 var numberError =cument.getElementById("numberError");
  numberError.innerHTML = "";

if(isNaN(number))
{
 numberError.innerHTML = "Please enter a valid number.";
 return false;
}
 return true;
 }
     </script>
   </body>
</html>
```

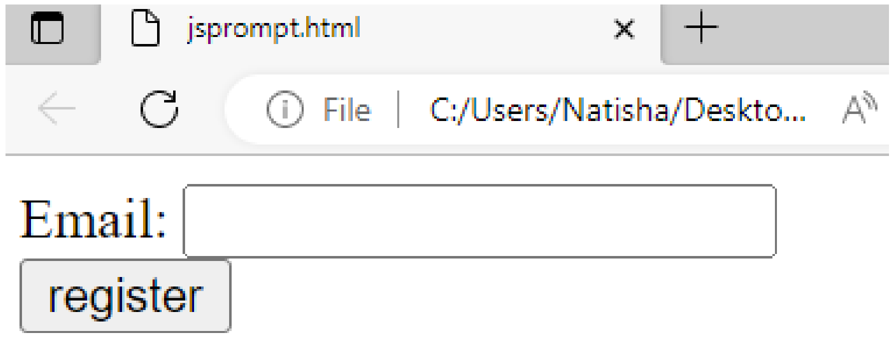
FORM VALIDATION- number validation

JavaScript email validation:

We can validate the email by the help of JavaScript.

- email id must contain the @ and . character
- There must be at least one character before and after the @.
- There must be at least two characters after . (dot).

```
<html> <body> <script>
function validateemail()
{
var x=document.myform.email.value;
var atposition=x.indexOf("@");
var dotposition=x.lastIndexOf(".");
if (atposition<1 || dotposition<atposition+2
|| dotposition+2>=x.length){
  alert("Please enter a valid e-mail address
\n atpostion:"+atposition+"\n
dotposition:"+dotposition);
  return false;
  }
}
</script>
<body>
<form name="myform"  method="post"
action="valid.html" onsubmit="return
validateemail();">
Email: <input type="text" name="email"><br/>

<input type="submit" value="register">
</form>
</body>
</html>
```


jsprompt.html
File
C:/Users/Natisha/Deskto...
Email:
register