*Experiment No.*_____ *Date___/___/2020*

TITLE OF EXPERIMENT: - A Program to illustrate different Set operations like-

- **Union**
- **Intersection**
- **Difference**
- **Set Difference**

DIVISION:_____ **BRANCH:** _____

BATCH:_____ **ROLL NO.:** _____

PERFORMED ON DATE: _____

SIGNATURE OF TEACHING STAFF:

EXPERIMENT NO. 7

Aim: Write a JavaScript program to illustrate different Set operations like-
• Union
• Intersection
• Difference
• Set Difference

Prerequisites:

- For this **Javascript Lab**, it is assumed that you have a prior knowledge of HTML coding. It would help if you had some prior exposure to object-oriented programming concepts and a general idea on creating online applications.
- To understand this experiment, you should have the knowledge of the basic **JavaScript, for loop, arithmetic operators, relational operators,**

- JavaScript Set and WeakSet
- JavaScript for... of Loop
- JavaScript Function and Function Expressions

Editor:

1.	NotePad
2.	Visual studio code

Theory:

Examples of different Set operations:

Input: A = {0, 2, 4, 6, 8}

```
B = {1, 2, 3, 4, 5}


Output:
 Union: [0, 1, 2, 3, 4, 5, 6, 8]
 Intersection: [2, 4]
 Difference: [8, 0, 6]
 Symmetric difference: [0, 1, 3, 5, 6, 8]
```

Union

Union (a ∪ b): create a set that contains the elements of both set a and set b.

```
let a = new Set([1,2,3]);
let b = new Set([4,3,2]);
let union = new Set([...a, ...b]);
   // {1,2,3,4}
```

The pattern is always the same:

- Convert one or both sets to arrays.
- Perform the operation.
- Convert the result back to a set.

As explained in above, the spread operator (...) inserts the elements of something iterable (like a set) into an array. Therefore, [...a, ...b] means that a and b are converted to arrays and concatenated. It is equivalent to [...a].concat([...b]).

Intersection

Intersection (a ∩ b): create a set that contains those elements of set a that are also in set b.

```
let a = new Set([1,2,3]);
let b = new Set([4,3,2]);
let intersection = new Set(
   [...a].filter(x => b.has(x)));
   // {2,3}
```

Steps: Convert a to an array, filter the elements, convert the result to a set.

Difference

Difference (a \ b): create a set that contains those elements of set a that are not in set b. This operation is also sometimes called *minus* (-).

```
let a = new Set([1,2,3]);
let b = new Set([4,3,2]);
let difference = new Set(
   [...a].filter(x => !b.has(x)));
   // {1}
```

Example 1: Set Union Operation

```
// perform union operation
// contain elements of both sets
function union(a, b) {
   let unionSet = new Set(a);
   for (let i of b) {
      unionSet.add(i);
   }
   return unionSet
}

// two sets of fruits
const setA = new Set(['apple', 'mango', 'orange']);
const setB = new Set(['grapes', 'apple', 'banana']);

const result = union(setA, setB);

console.log(result);
Run Code
```

Output

```
Set {"apple", "mango", "orange", "grapes", "banana"}
```

The set union operation combines elements of both sets into one.

A new set unionSet is created using new Set(). The unionSet variable contains all the values of setA. Then, the for...of loop is used to iterate through all the elements of setB and add them to unionSet using the add() method.

The set does not contain duplicate values. Hence, if the set contains the same value, the latter value is discarded.

Example 2: Set Intersection Operation

```
// perform intersection operation
```

```
// elements of set a that are also in set b

function intersection(setA, setB) {

    let intersectionSet = new Set();

    for (let i of setB) {

        if (setA.has(i)) {

            intersectionSet.add(i);

        }

    }

    return intersectionSet;

}



// two sets of fruits

const setA = new Set(['apple', 'mango', 'orange']);

const setB = new Set(['grapes', 'apple', 'banana']);

const result = intersection(setA, setB);

console.log(result);
```

Output

```
Set {"apple"}
```

The set intersection operation represents elements that are present in both setA and setB.

A new set intersectionSet is created using new Set(). Then, the for...of loop is used to iterate through the setB. For every element that is present in both setA and setB, they are added to the intersection set.

Example 3: Set Difference Operation

```
// perform difference operation
// elements of set a that are not in set b
function difference(setA, setB) {
   let differenceSet = new Set(setA)
   for (let i of setB) {
      differenceSet.delete(i)
   }
   return differenceSet
}

// two sets of fruits
const setA = new Set(['apple', 'mango', 'orange']);
const setB = new Set(['grapes', 'apple', 'banana']);

const result = difference(setA, setB);

console.log(result);
Run Code
```

Output

```
Set {"mango", "orange"}
```

The set difference operation represents elements that are present in one set and not in another set.

The differenceSet contains all the elements of setA. Then, the for...of loop is used to iterate through all the elements of setB. If the element that is present in setB is also available in setA, that element is deleted using delete() method.

Example 4: Set Subset Operation

```
// perform subset operation
// true if all elements of set b is in set a
function subset(setA, setB) {
    for (let i of setB) {
        if (!setA.has(i)) {
            return false
        }
    }
    return true
}

// two sets of fruits
const setA = new Set(['apple', 'mango', 'orange']);
const setB = new Set(['apple', 'orange']);

const result = subset(setA, setB);

console.log(result);
```

Output

```
true
```

The set subset operation returns true if all the elements of setB are in setA.

The for...of loop is used to loop through the elements of setB. If any element that is present is setB is not present in setA, false is returned.

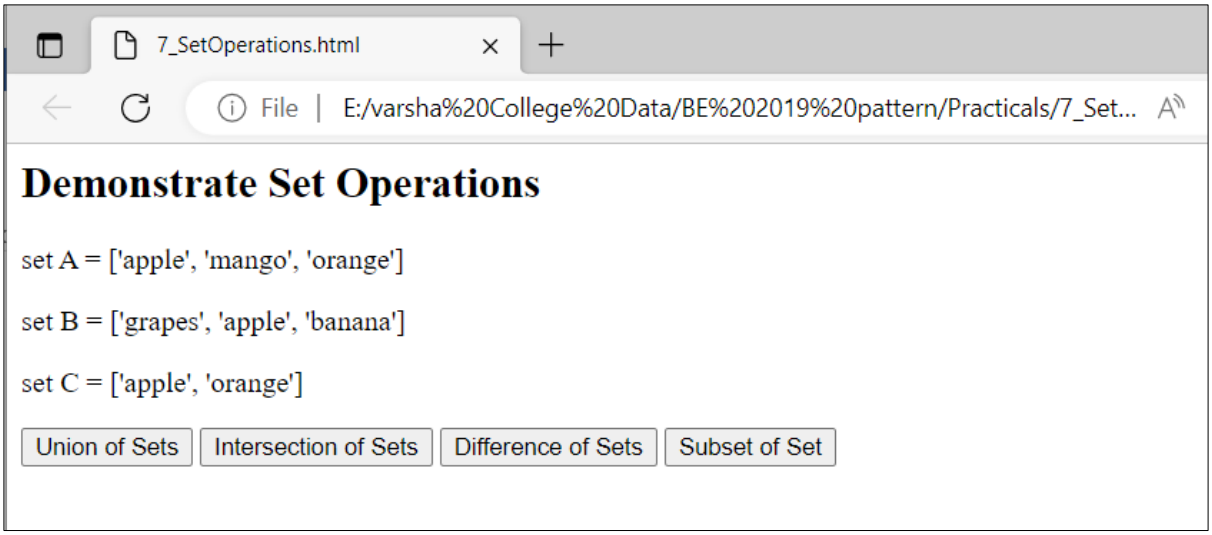
Program:

```
<html>
<head>
<h2> Demonstrate Set Operations </h2>
<p> set A = ['apple', 'mango', 'orange'] </p>
<p> set B = ['grapes', 'apple', 'banana'] </p>
<p> set C = ['apple', 'orange'] </p>
<button onclick="union()">Union of Sets</button>
<button onclick="intersection()">Intersection of Sets</button>
```

```
<button onclick="difference()">Difference of Sets</button>
<button onclick="subset()">Subset of Set</button>
<script>

// two sets of fruits
const setA = new Set(['apple', 'mango', 'orange']);
const setB = new Set(['grapes', 'apple', 'banana']);
const setC = new Set(['apple', 'orange']);

function union() {
    let unionSet = new Set(setA);
    for (let i of setB) {
        unionSet.add(i);
    }
console.log(unionSet);

}

function intersection() {
    let intersectionSet = new Set();

    for (let i of setB) {
        if (setA.has(i)) {
            intersectionSet.add(i);
        }
    }
   console.log(intersectionSet);
}

function difference() {
```

```
    let differenceSet = new Set(setA)
    for (let i of setB) {
        differenceSet.delete(i)
    }
console.log(differenceSet);
}

function subset() {
    for (let i of setC) {
        if (!setA.has(i)) {
            console.log(false);
        }
    }


 console.log(true);
}



</script>
</head>
</html>
```

Screenshot’s of Output:



Conclusion: