Remove elements from array in JavaScript

An array is a variable used to store one or more elements of the same data type. Basically, it stores multiple elements of the same type. Sometimes we need to remove these elements from an array. JavaScript offers several built-in array methods to add or remove the elements from an array easily. Using these methods, you can remove an element from start, end, or as well as from a specific index.

These JavaScript array methods are as follows:

Method	Description
pop()	This method removes the elements from the end of the array.
shift()	Like the pop() method, it also removes the elements but from the start of the array.
filter()	The filter() method removes the elements from an array in a programmatically way.
splice()	This method removes the elements from a specific index.

All the above methods are array functions offered by JavaScript. These methods are discussed below in detail with examples.

Remove elements from the end of the array - pop()

JavaScript provides the pop() method to remove the elements from the end of the a
and returns the removed element. When an element removes from the array, the leng
and output below to understand:

Example 1

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

  function removeLastElement() {
   var shoeBrand = ["Nike", " Adidas", " Sparks", " RedTape"];

   document.write("Elements in array before removing: <br>" + shoeBrand + "<br> <br>");

   // Removing last element from the array
   var poppedElement = shoeBrand.pop();
   document.write("Removed element from array: " + poppedElement + "<br> <br

                  ng elements present in array after removing
                  "Elements present in array: <br>" + shoeBrand);
```

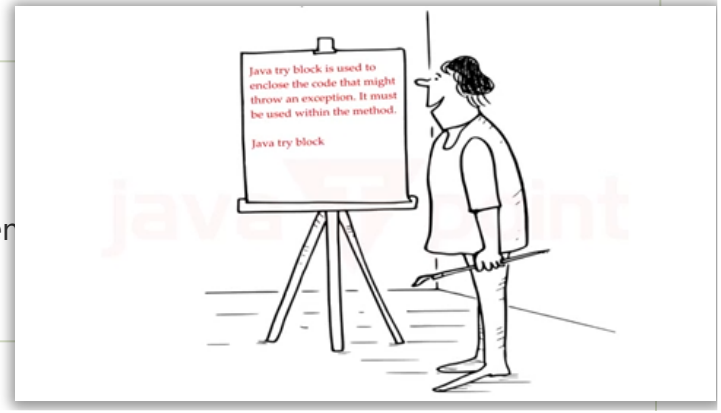
```
}
removeLastElement();

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

Test it Now

Output

Initially, there are four elements in the array. One element from the last will be removed using the pop() function and three elements will remain in that array.

```
Elements in array before removing:
Nike, Adidas, Sparks, RedTape

Removed element from array: RedTape

Elements present in array:
Nike, Adidas, Sparks
```



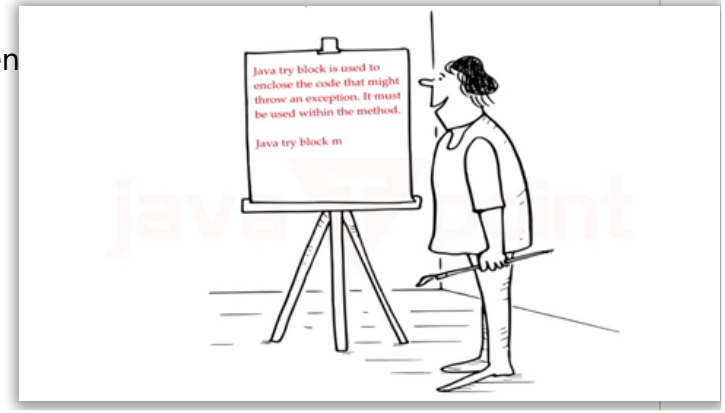
Example 2

By putting the above code in a loop (for, while, or do-while), we can delete all elemer
how it will work:

⇧ SCROLL TO TOP

```
<html>
<body>
<script>
  function removeElement() {
   var shoeBrand = ["Nike", " Adidas", " Sparks", " RedTape"];

   //initial length of the array
   document.write("Elements in array before removing: <br>" + shoeBrand + "<br> <br>");
   document.write("Array length before removing elements is:" + shoeBrand.length + "<br> <br>");

   while (shoeBrand.length) {
   //store removed element in a variable
   var poppedElement = shoeBrand.pop();

   //display removed element
   document.write("Removed element from array: " + poppedElement + " <br>");
   }

   //Length of the array after removing all elements
   document.write("<br> Array length after removing elements is:" + shoeBrand.len

}
removeElement();
</script>
```

⇧ SCROLL TO TOP



```
</html>
```

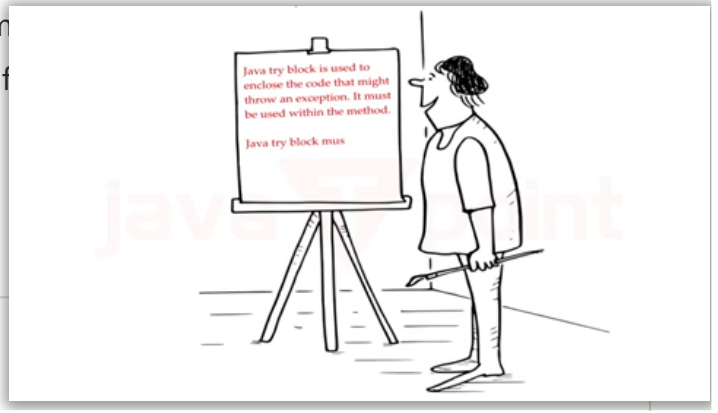
Test it Now

Output

```
Elements in array before removing:
Nike, Adidas, Sparks, RedTape

Array Length after removing elements is: 4

Removed element from array: RedTape
Removed element from array: Sparks
Removed element from array: Adidas
Removed element from array: Nike

Array Length after removing elements is: 0
```

Remove elements from the start of the array - shift()

JavaScript provides the shift() method, which is used to remove the element from
element from an array and returns the removed element. When an element removes f
1. See the code and output below how this function works:



Example 1

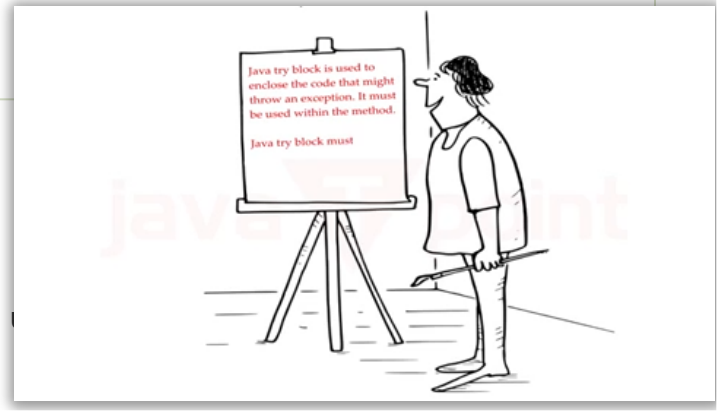
⇧ SCROLL TO TOP

```
<body>
<script>

  function removeFirstElement() {
   var shoeBrand = ["Nike", " Adidas", " Sparks", " RedTape"];

   document.write("Elements in array before removing: <br>" + shoeBrand + "<br> <br>");

   // Removing first element from the array
   var poppedElement = shoeBrand.shift();
   document.write("Removed element from array: " + poppedElement + "<br> <br>");

   //display remaining elements present in array after removing
   document.write("Elements present in array: <br>" + shoeBrand);
}
removeFirstElement();

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

Test it Now

Output

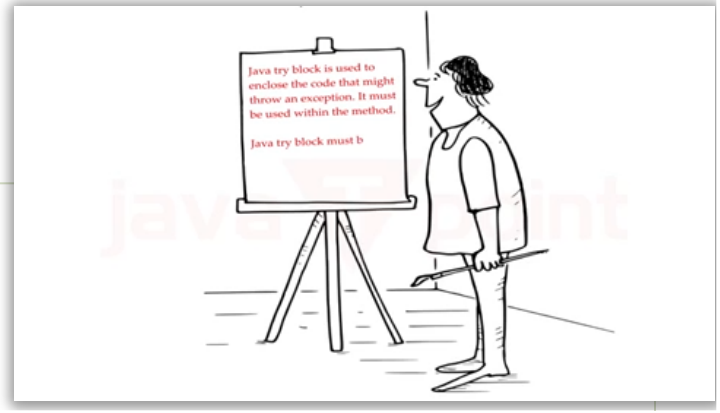
r elements in the array. One element from the start will remove
ay.



```
Elements in array before removing:
Nike, Adidas, Sparks, RedTape

Removed element from array: Nike

Elements present in array:
Adidas, Sparks, RedTape
```

Example 2

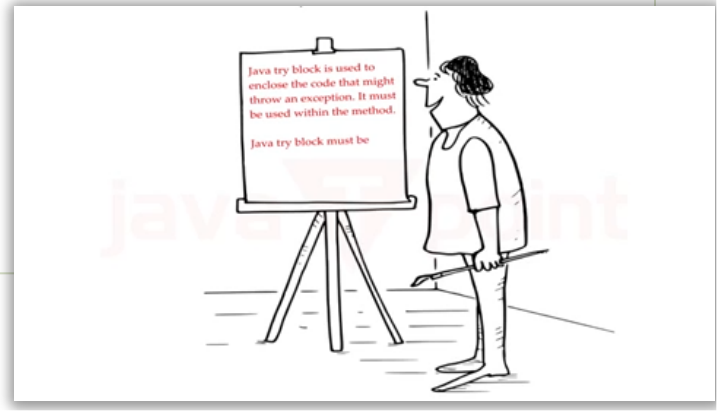
Like the pop() method, we can delete all elements one by one from the start of the array by putting the above code in a loop (for, while, or do-while). In this example, we will put this code in a while loop. See how it will work:




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

⇧ SCROLL TO TOP

```
function removeElement() {
 var shoeBrand = ["Nike", " Adidas", " Sparks", " RedTape"];

 //initial length of the array
 document.write("Elements in array before removing: <br>" + shoeBrand + "<br> <br>");
 document.write("Array length before removing elements is:" + shoeBrand.length + "<br> <br>");

 while (shoeBrand.length) {
 //store removed element in a variable
 var poppedElement = shoeBrand.shift();

 //display removed element
 document.write("Removed element from array: " + poppedElement + " <br>");
 }

 //Length of the array after removing all elements
 document.write("<br> Array length after removing elements is:" + shoeBrand.length);

}
removeElement();
</script>
</body>
</html>
```



⇧ SCROLL TO TOP

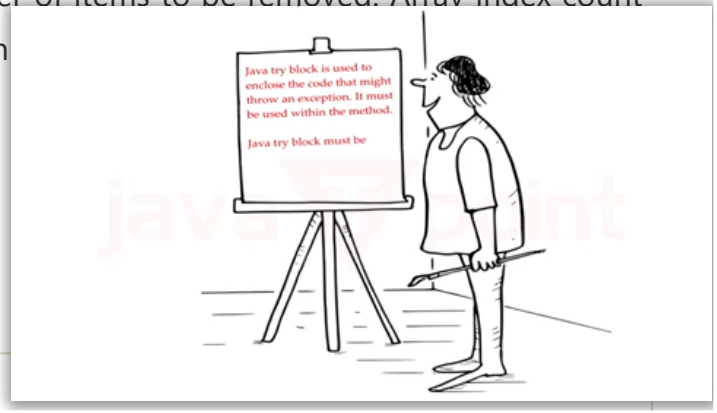
Output

```
Elements in array before removing:
Nike, Adidas, Sparks, RedTape

Array Length after removing elements is: 4

Removed element from array: Nike
Removed element from array: Adidas
Removed element from array: Sparks
Removed element from array: RedTape

Array Length after removing elements is: 0
```

Remove elements from a specific index in an array - splice()

To remove the element from a specific index position, the splice() method is used. It removes the element from a specific position and returns that removed element. It also allows the users to remove one or more elements from the array.

The splice() method accepts mainly two arguments: initial index position and number of items to be removed. Array index count starts from 0, i.e., a[0]. When the elements remove from an array, the array length output how the splice() function works:



Example 1

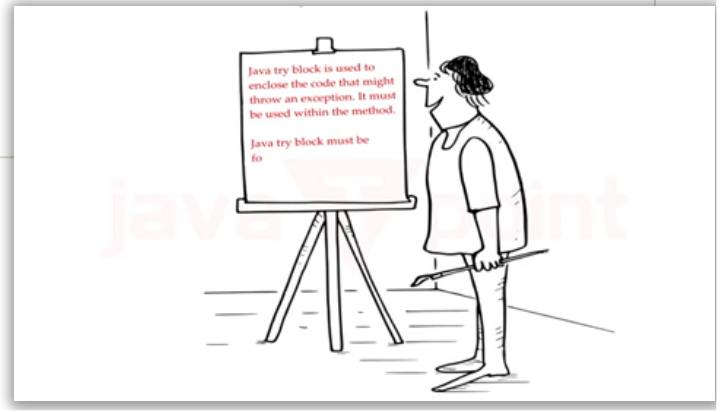
In this example, we will delete three elements, starts from index 1, i.e., a[1] to a[3].

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

  function removeElement() {
   var shoeBrand = ["Nike", " Adidas", " Sparks", " RedTape", " Bata"];

   document.write("Elements in array before removing: <br>" + shoeBrand + "<br> <br>");

   // Removing first element from the array
   var poppedElement = shoeBrand.splice(1, 3);
   document.write("Removed element from array: " + poppedElement + "<br> <br>");

   //display remaining elements present in array after removing
   document.write("Elements present in array: <br>" + shoeBrand);
}
removeElement();

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

Test it Now

Output

⇧ SCROLL TO TOP

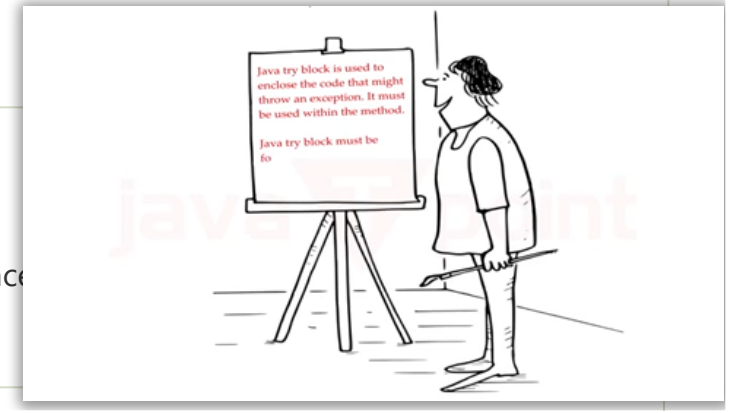


In the below response, you can see that three elements from the array have been removed, and only two elements (Nike and Bata) have remained in the array.

```
Elements in array before removing:
Nike, Adidas, Sparks, RedTape, Bata

Removed element from array: Adidas, Sparks, RedTape,

Elements present in array:
Nike, Bata
```

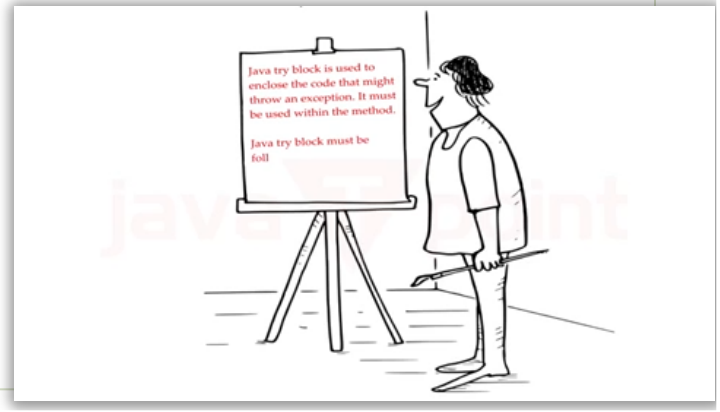
Example 2

In this example, we will put the above code inside a for loop to remove all occurrence
and remove the matching element one by one from the array.



```
<html>
<body>

<script>
function removeElement() {
   var clothingBrand = ["Gucci", " Chanel", "Gucci", " Zara"];

   // for loop to trace the whole array
   for (var i = 0; i < clothingBrand.length; i++) {

      //Match the specific element in array
      if (clothingBrand[i] === "Gucci") {
         //remove the matched element from array
         var delEle = clothingBrand.splice(i, 1);

         document.write("<br> Removed element: " + delEle);
         document.write("<br> Remaining elements: " + clothingBrand);
         document.write("<br>");       }
   }
}
removeElement();
</script>

</body>
```

⇧ SCROLL TO TOP



Test it Now

Output

You can see that element named (**Gucci**) has been removed twice from the array in the below output, and only two elements (Chanel, Zara) have remained in the array.

```
Removed element: Gucci
Remaining Element: Chanel, Gucci, Zara

Removed element: Gucci
Remaining Element: Chanel, Zara
```

You can even remove all elements from the array. See the below code:

```
<script>

    var clothingBrand = ["Gucci", " Chanel", " Calvin Klein", " Zara"];
    document.write("Elements in array: " + clothingBrand);
    //remove all elements
    clothingBrand.splice(0, clothingBrand.length);
    document.write("<br> Remaining elements: " + clothingBrand);
```


```
</script>
```

Output

See that all elements have been deleted.

```
Elements in array: Gucci, Chanel, Calvin Klein, Zara
Remaining Element:
```

Remove elements from the array using filter()

This method basically removes the element based on the given condition provided by the user. It removes the elements and creates a new array of remaining elements. See the code and output below how it works:

Example 1

In this example, we will check the even-odd values in an array and filter them. The filter() method will check for the even values and return to add them to the modified array. The odd values will remove from the array, and only modified array will be displayed.

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

function isEven( value ) {
```

```
}
    //initialize the array named ary
    var ary = [43, 243, 56, 24, 1021, 348].filter( isEven );
    document.write("Even elements in array: " + ary);

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

Test it Now

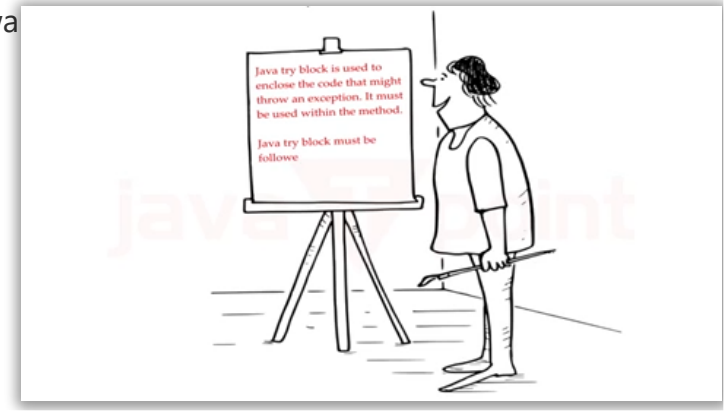
Output

See the output below that only even elements have remained in the modified array:

```
Even elements in array:  56, 24, 348
```

Remove elements using delete operator

Apart from all these functions, JavaScript offers a **delete** operator. It helps to remove the element from a specific index position in an array. This operator is used with array name and index number, which you wa returns true after successfully removing an element.



⇧ SCROLL TO TOP

The **delete** operator helps to remove specific index element directly from the array. Now, with the help of an example, let us see how this **delete** operator works:

Example

```
<html>
<body>

<script>
        //declare and initialize an array
        var clothingBrand = ["Gucci", " Calvin Klein", " Chanel", " Zara"];
        document.write("Elements in array: " + clothingBrand);

        //delete element of index 1 from clothingBrand array
        delete clothingBrand[1];
```


```
        //if returned value is true, element is deleted successfully
        document.write("<br> Removed successfully: " + result + "<br>");
        document.write("Remaining elements in array: " + clothingBrand);
</script>

</body>
</html>
```

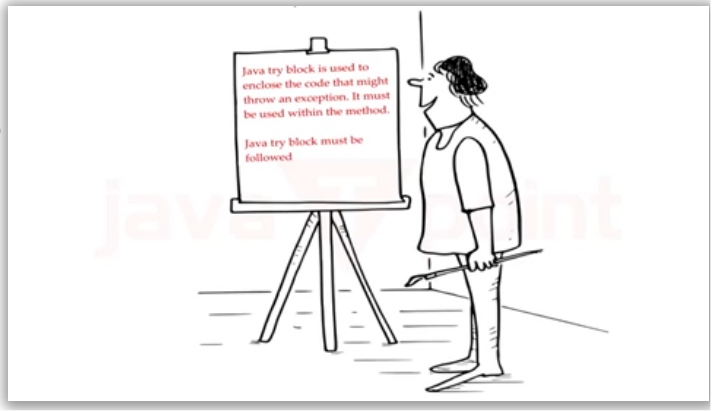
Test it Now

Output

In this output, you can see that if the returned value is **true** after performing the remove operation, the element presents at index 1 has been deleted successfully.

```
Elements in array:  Gucci, Calvin Klein, Chanel, Zara
Removed successfully: true
Remaining elements in array: Gucci,, Chanel, Zara
```

Remove elements using clear and reset operator



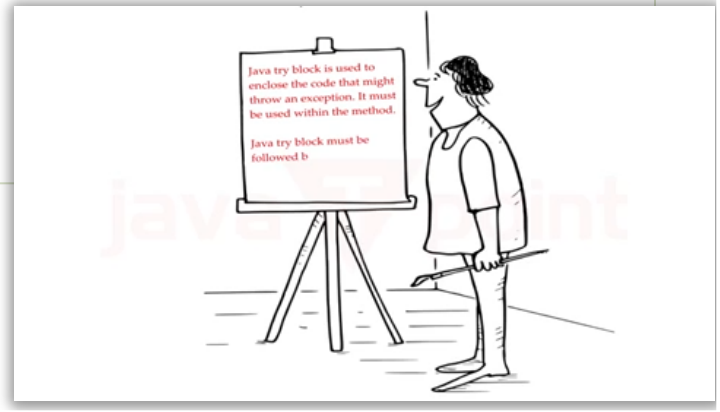
JavaScript provides the **clear** and **reset** operator to remove the elements from the elements; they just shift them to another array and clear the original array.

Now, with the help of an example, let us see how it works:

⇧ SCROLL TO TOP

```
<html>
<body>

<script>
        //declare and initialize an array
        var originalArray = ["Gucci", " Calvin Klein", " Chanel", " Zara"];
        document.write("Initially elements in array: " + originalArray);

        //declare one more array to keep the elements of original array
        var newArray = originalArray

        //clear the initially declared array
        originalArray = [ ]

        //display element of original and new array after removing
        document.write("<br> <br> Array after removing elements: " + originalArray);
        document.write("<br> <br> Elements in new array: " + newArray);
</script>

</body>
</html>
```

Test it Now



⇧ SCROLL TO TOP

In this output, you can see that the original array elements have been shifted to a new array. The initially declared array has been empty, which means no element present in array now.

```
Initially elements in array:  Gucci, Calvin Klein, Chanel, Zara

Array after removing elements:

Elements in new array: Gucci, Calvin Klein, Chanel, Zara
```

Example 2

Other than this, we can remove all elements of the array by setting its length to 0. See the example below:

```
<html>
<body>

<script>
        //declare and initialize an array
        var array1 = ["Gucci", " Calvin Klein", " Chanel", " Zara"];
        document.write("Initially elements in array: " + array1);

        //set length of array to 0
        array1.length = 0;

                                ıent of original and new array after removing
                            .ıte("<br> <br> Array after removing elements: " + array1);
```

```
</script>

</body>
</html>
```

Test it Now

Output

By setting the array length to 0, all elements of the array have been disabled or removed. See the empty array:

```
Initially elements in array:  Gucci, Calvin Klein, Chanel, Zara

Array after removing elements:
```

← Prev

Next →

⇧ SCROLL TO TOP