JavaScript TUTORIAL

Remove Duplicates from an Array

Summary: in this tutorial, you will learn how to remove duplicates from an array in JavaScript.

1) Remove duplicates from an array using a Set

A `Set` (https://www.javascripttutorial.net/es6/javascript-set/) is a collection of unique values. To remove duplicates from an array (https://www.javascripttutorial.net/javascript-array/) :

- First, convert an array of duplicates to a `Set`. The new `Set` will implicitly remove duplicate elements.
- Then, convert the `set` back to an array.

The following example uses a `Set` to remove duplicates from an array:

```
let chars = ['A', 'B', 'A', 'C', 'B'];
let uniqueChars = [...new Set(chars)];

console.log(uniqueChars);
```

Output:

```
[ 'A', 'B', 'C' ]
```

2) Remove duplicates from an array using indexOf() and filter() methods

The `indexOf()` (https://www.javascripttutorial.net/javascript-array-indexof/) method returns the index of the first occurrence of an element in an array. For example:

```
let chars = ['A', 'B', 'A', 'C', 'B'];
chars.indexOf('B');
```

Output:

```
1
```

The duplicate item is the item whose index is different from its `indexOf()` value:

```
let chars = ['A', 'B', 'A', 'C', 'B'];

chars.forEach((c, index) => {
    console.log(`${c} - ${index} - ${chars.indexOf(c)}`);
});
```

Output:

```
A - 0 - 0
B - 1 - 1
A - 2 - 0
C - 3 - 3
B - 4 - 1
```

To remove the duplicates, you use the `filter()` (https://www.javascripttutorial.net/javascript-array-filter/) method to include only elements whose indexes match their indexOf values:

```
let chars = ['A', 'B', 'A', 'C', 'B'];

let uniqueChars = chars.filter((c, index) => {
    return chars.indexOf(c) === index;
});

console.log(uniqueChars);
```

Output:

```
[ 'A', 'B', 'C' ]
```

To find the duplicate values, you need to reverse the condition:

```
let chars = ['A', 'B', 'A', 'C', 'B'];

let dupChars = chars.filter((c, index) => {
    return chars.indexOf(c) !== index;
});

console.log(dupChars);
```

Output:

```
[ 'A', 'B' ]
```

3) Remove duplicates from an array using forEach() and include()

The `include()` (https://www.javascripttutorial.net/es6/javascript-string-includes/) returns `true` if an element is in an array or `false` if it is not.

The following example iterates over elements of an array and adds to a new array only elements that are not already there:

```
let chars = ['A', 'B', 'A', 'C', 'B'];

let uniqueChars = [];
chars.forEach((c) => {
    if (!uniqueChars.includes(c)) {
        uniqueChars.push(c);
    }
});
```

```
console.log(uniqueChars);
```

Output:

```
[ 'A', 'B', 'C' ]
```

4) Remove duplicates from an array of objects by one property

Suppose you have the following array of objects:

```
const members = [
  { id: 1, name: 'John' },
  { id: 2, name: 'Jane' },
  { id: 1, name: 'Johnny' },
  { id: 4, name: 'Alice' },
];
```

The id of the first is the same as the third element. To remove the duplicate from the people array, you can use the following:

```
const unique = [...new Map(members.map((m) => [m.id, m])).values()];
console.log(unique);
```

Output:

```
[
  { id: 1, name: 'Johnny' },
  { id: 2, name: 'Jane' },
  { id: 4, name: 'Alice' }
]
```

How it works.

First, create a new array from the original array using the `map()` (https://www.javascripttutorial.net/javascript-array-map/) method:

```
members.map((m) => [m.id, m])
```

It returns an array of arrays. Each nested array contains the value of the id and the corresponding object:

```
[
  [ 1, { id: 1, name: 'John' } ],
  [ 2, { id: 2, name: 'Jane' } ],
  [ 1, { id: 1, name: 'Johnny' } ],
  [ 4, { id: 4, name: 'Alice' } ]
]
```

Second, remove the duplicate by creating a new Map() object:

```
const newMap = new Map(newArray);
console.log(newMap);
```

Output:

```
Map(3) {
  1 => { id: 1, name: 'Johnny' },
  2 => { id: 2, name: 'Jane' },
  4 => { id: 4, name: 'Alice' }
}
```

Because the keys of a Map object are unique, creating a Map from the array of array removes the duplicate object by key (id in this case).

Third, get the iterator of the Map's entries by calling the values() method:

```
const iterator = newMap.values();
console.log(iterator);
```

Output:

```
[Map Iterator] {
  { id: 1, name: 'Johnny' },
  { id: 2, name: 'Jane' },
  { id: 4, name: 'Alice' }
}
```

Finally, convert the iterator to an array by using the spread operator:

```
const uniqueMembers = [...iterator];
console.log(uniqueMembers);
```

Output:

```
[
  { id: 1, name: 'Johnny' },
  { id: 2, name: 'Jane' },
  { id: 4, name: 'Alice' }
]
```

Put it all together:

```
const members = [
  { id: 1, name: 'John' },
  { id: 2, name: 'Jane' },
  { id: 1, name: 'Johnny' },
  { id: 4, name: 'Alice' },
];

const newArray = members.map((m) => [m.id, m]);
const newMap = new Map(newArray);
const iterator = newMap.values();
const unique = [...iterator];
```

```
console.log(unique);
```

The following four lines of code:

```
const newArray = members.map((m) => [m.id, m]);
const newMap = new Map(newArray);
const iterator = newMap.values();
const unique = [...iterator];
```

...can be shorted into one:

```
const unique = [...new Map(members.map((m) => [m.id, m])).values()];
```

So:

```
const members = [
  { id: 1, name: 'John' },
  { id: 2, name: 'Jane' },
  { id: 1, name: 'Johnny' },
  { id: 4, name: 'Alice' },
];

const unique = [...new Map(members.map((m) => [m.id, m])).values()];

console.log(unique);
```

The following unique() function accepts an array of objects and returns the unique element by a property:

```
const uniqueBy = (arr, prop) => {
  return [...new Map(arr.map((m) => [m[prop], m])).values()];
};
```

For example, you can use the `uniqueBy()` function to remove duplicate elements from the `members` array like this:

```
const members = [
  { id: 1, name: 'John' },
  { id: 2, name: 'Jane' },
  { id: 1, name: 'Johnny' },
  { id: 4, name: 'Alice' },
];

const uniqueBy = (arr, prop) => {
  return [...new Map(arr.map((m) => [m[prop], m])).values()];
};

console.log(uniqueBy(members, 'id'));
```

5) Remove duplicates from an array of objects by multiple properties

The following unique() funciton remove duplicate from an array of object. The duplicate logic is specified by a callback function (https://www.javascripttutorial.net/javascript-callback/) :

```
function unique(a, fn) {
  if (a.length === 0 || a.length === 1) {
    return a;
  }
  if (!fn) {
    return a;
  }

  for (let i = 0; i < a.length; i++) {
    for (let j = i + 1; j < a.length; j++) {
      if (fn(a[i], a[j])) {
        a.splice(i, 1);
      }
    }
  }
  return a;
}
```

How it works.

First, return the same array if it has zero or one element:

```
if (a.length === 0 || a.length === 1) {
  return a;
}
```

Second, return the input array if the callback is not passed:

```
if (!fn) {
  return a;
}
```

Third, iterate over the element of the input array twice and successively compare the first element with the other elements. If two elements cause the callback function (`fn`) to return true, remove that element from the array using the `splice()` (https://www.javascripttutorial.net/javascript-array-splice/) method.

```
for (let i = 0; i < a.length; i++) {
  for (let j = i + 1; j < a.length; j++) {
    if (fn(a[i], a[j])) {
      a.splice(i, 1);
    }
  }
}
```

The following example uses the unique() function to remove duplicates from the `members` array by both `id` and `name` properties:

```
function unique(a, fn) {
  if (a.length === 0 || a.length === 1) {
    return a;
  }
  if (!fn) {
    return a;
  }

  for (let i = 0; i < a.length; i++) {
    for (let j = i + 1; j < a.length; j++) {
      if (fn(a[i], a[j])) {
        a.splice(i, 1);
      }
    }
  }
  return a;
}

const members = [
  { id: 1, name: 'John' },
  { id: 2, name: 'Jane' },
  { id: 1, name: 'John' },
  { id: 4, name: 'Joe' },
];

const uniqueMembers = unique(
  members,
  (a, b) => (a.id === b.id) & (a.name === b.name)
);

console.log(uniqueMembers);
```

Output:

```
[
  { id: 2, name: 'Jane' },
  { id: 1, name: 'John' },
  { id: 4, name: 'Joe' }
]
```

In this tutorial, you have learned some techniques to remove duplicates from an array in JavaScript.