JavaScript Array Length

Summary: in this tutorial, you'll learn about the JavaScript Array `length` property and how to handle it correctly.

What exactly is the JavaScript Array length property

By definition, the `length` property of an array is an **unsigned, 32-bit integer** that is always **numerically greater than the highest index** in the array.

The maximum value of the `length` is 2^{32}. This means that an array can hold up to `4_294_967_296` (2^{32}) elements.

The `length` property behaves differently depending on the array types including dense and sparse.

1) Dense arrays

A dense array is an array where its elements have contiguous indexes starting at zero.

For dense arrays, you can use the `length` property to get the number of elements in the array. For example:

```
let colors = ['red', 'green', 'blue'];
console.log(colors.length); // 3
```

In this example, the `length` property returns three, which is the same as the number of elements in the `colors` array.

The following adds one more element to the `colors` array:

```
let colors = ['red', 'green', 'blue'];
console.log(colors.length); // 3
```

```
colors.push('yellow');
console.log(colors.length); // 4
```

Now, the `length` property of the `colors` array is 4.

When you empty the `colors` array, its length is zero:

```
colors = [];
console.log(colors.length); // 0
```

2) Sparse arrays

A sparse array is an array whose elements don't have contiguous indexes starting at zero.

For example, the `[10,, 20, 30]` is a sparse array because the indexes of its elements are 0, 2, and 3.

In a sparse array, the `length` property doesn't indicate the actual number of elements. It's a number that is greater than the highest index. For example:

```
let numbers = [10, , 20, 30];
console.log(numbers.length); // 4
```

In this example, the number of elements in the `numbers` array is 3: `10` , `20` , and `30` . The highest index is 3. Therefore, the `length` property returns `4` .

The following adds an element to the `numbers` array at the index 10:

```
let numbers = [10, , 20, 30];
console.log(numbers.length); // 4

numbers[10] = 100;
console.log(numbers.length); // 11
```

In this example, the `length` property returns 11.

Modifying JavaScript Array length property

JavaScript allows you to change the value of the array `length` property. By changing the value of the `length` property, you can remove elements from an array or make an array sparse.

1) Emptying an array

If you set the `length` property of an array to zero, the array will be empty:

```
const fruits = ['Apple', 'Orange', 'Strawberry'];
fruits.length = 0;

console.log(fruits); // []
```

Output:

```
[]
```

2) Removing elements

If you set the `length` property of an array to a value lower than the highest index, all the elements whose index is greater than or equal to the new length are removed.

The following example changes the `length` property of the `fruits` array to 2, which removes the third element from the array:

```
const fruits = ['Apple', 'Orange', 'Strawberry'];
fruits.length = 2;

console.log(fruits);
```

Output:

```
[ 'Apple', 'Orange' ]
```

3) Making the array sparse

If you set the `length` property of an array to a value higher than the highest index, the array will be spared. For example:

```
const fruits = ['Apple', 'Orange', 'Strawberry'];
fruits.length = 5;

console.log(fruits);
```

Output:

```
[ 'Apple', 'Orange', 'Strawberry', <2 empty items> ]
```

Summary

- The `length` property of an array is an unsigned, 32-bit integer that is always numerically greater than the highest index of the array.
- The `length` property returns the number of elements of a dense array.
- For spare arrays, the `length` property doesn't reflect the number of elements.
- Modifying the `length` property may remove elements from an array or make an array spare.