Array.prototype.pop()

Summary: in this tutorial, you'll learn how to use the JavaScript Array `pop()` method to remove the last element from an array.

Introduction to the JavaScript Array pop() method

The `Array.prototype.pop()` method removes the last element from an array and returns the removed element.

Here's the syntax of the `pop()` method:

```
const e = array.pop()
```

The `pop()` method modifies the original array by changing its `length` property. If the array is empty, the `pop()` returns `undefined`.

The `pop()` method modifies the original array. To remove one or more elements from the end of an array without changing the original array, you can use the slice() method:

```
const newArray = array.slice(0, array.length - elementsToRemove);
```

The `slice()` method removes the number of elements (`elementsToRemove`) from the array and returns a new array (`newArray`).

JavaScript pop() method examples

Let's take some examples of using the `pop()` method.

1) Removing the last element of an array

The following example uses the `pop()` method to remove the last element of the `numbers` array:

```
const numbers = [10, 20, 30];
const last = numbers.pop();

console.log({ last });
console.log({ length: numbers.length });
```

Output:

```
{ last: 30 }
{ length: 2 }
```

In this example, the `pop()` method removes the number `30` from the `numbers` array. Also, it decreases the value of the `length` property of the `numbers` array to `2` .

The following picture illustrates how the `pop()` method works:

2) Calling the pop() method on an empty array

The following example calls the `pop()` method on an empty array. In this case, the `pop()` method returns `undefined` and the `length` is of the array is zero:

```
const numbers = [];
const last = numbers.pop();

console.log({ last });
console.log({ length: numbers.length });
```

Output:

```
{ last: undefined }
{ length: 0 }
```

Using the pop() method with array-like objects

The `pop()` method is generic, meaning that you can use the `call()` or `apply()` to call the `pop()` method on an array-like object.

Internally, the `pop()` uses the `length` property of the array-like object to determine the last element to remove. For example:

```
let greetings = {
  0: 'Hi',
  1: 'Hello',
  2: 'Howdy',
  length: 3,
  removeLast() {
    return [].pop.call(this);
  },
};

let greting = greetings.removeLast();

console.log({ greting });
console.log(greetings);
```

Output:

```
{ greting: 'Howdy' }
{
  '0': 'Hi',
  '1': 'Hello',
  length: 2,
  removeLast: [Function: removeLast]
}
```

How it works.

First, define the `greetings` object that has:

- Four properties 0, 1, 2, and length.
- One method `removeLast()` that uses the `call()` method of an array to invoke the `pop()` method.

Second, call the `removeLast()` method of the `greetings` object:

```
let greting = greetings.removeLast();
```

Third, output the removed element (`greeting`) and the `greetings` object to the console:

```
console.log({ greting });
console.log(greetings);
```

Summary

- Use the JavaScript Array `pop()` method to remove the last element of an array.
- Use the `call()` or `apply()` to call the `pop()` method on an array-like object.