JavaScript TUTORIAL

Array.prototype.unshift()

Summary: in this tutorial, you'll learn how to use the JavaScript Array `unshift()` method to add one or more elements to the beginning of an array.

Introduction to the JavaScript Array unshift() method

The `unshift()` method adds one or more elements to the beginning of an array and returns the new array's length.

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

```
unshift(element);
unshift(element1, element2);
unshift(element1, element2,...elementN);
```

The `unshift()` method is slow for large arrays because it needs to reindex the existing elements.

> To add one or more elements to the end of an array, you can use the `push()` method instead.

Note that the `unshift()` method modifies the original array. If you want to prepend an element and get the new array without modifying the original array, you can use the following syntax:

```
const newArray = [newElement, ...array]
```

In this syntax, we create a new array by placing the new element first (`newElement`) and then the elements of the original array. The spread operator (`...`) spreads out the elements of the original array.

JavaScript Array unshift() method examples

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

1) Prepend an element to an array

The following example uses the `unshift()` method to add the number `10` to the beginning of the `numbers` array:

```
let numbers = [30, 40];

const length = numbers.unshift(20);

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

Output:

```
{ length: 3 }
{ numbers: [ 20, 30, 40 ] }
```

How it works.

First, define an array that includes two numbers:

```
let numbers = [20, 30];
```

The length of the `numbers` array is 2.

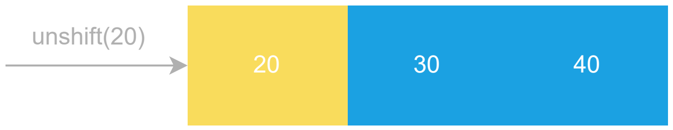
Second, add the number `10` to the beginning of the `numbers` array and assign the returned value to the `length` variable:

```
const length = numbers.unshift(10);
```

Third, output the `length` and `numbers` variables to the console:

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

The following picture illustrates how the `unshift()` function works:



2) Prepend multiple elements to an array

The following example uses the `unshift()` method to add two elements to the beginning of an array:

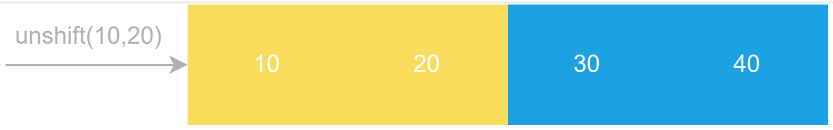
```
let numbers = [30, 40];

const length = numbers.unshift(10, 20);

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

Output:

```
{ length: 4 }
{ numbers: [ 10, 20, 30, 40 ] }
```



3) Prepend elements of an array to another array

The following example uses the `unshift()` method to prepend elements of an array to the beginning of another array:

```
const days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri'];
const weekends = ['Sat', 'Sun'];
```

```
for (const weekend of weekends) {
  days.unshift(weekend);
}

console.log(days);
```

Output:

```
['Sun', 'Sat', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri']
```

To make it more concise, you can use the spread operator like this:

```
let days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri'];
let weekends = ['Sat', 'Sun'];

days.unshift(...weekends);

console.log(days);
```

The spread operator ... spreads out the elements of the `weekends` array and the unshift() method prepends them to the `days` array. It looks like the following internally:

```
days.unshift('Sat','Sun');
```

Using the JavaScript Array unshift() method with array-like objects

The `unshift()` method is generic, meaning it can work well with array-like objects.

To call the `unshift()` method from an array-like object, you borrow it from an array object using the `call()` or `apply()` method. For example:

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

greetings.prepend('Good day');

console.log(greetings);
```

Output:

```
{
  '0': 'Good day',
  '1': 'Hi',
  '2': 'Hello',
  '3': 'Howdy',
  length: 4,
  prepend: [Function: prepend]
}
```

How it works.

First, define the `greetings` object that has

- The properties with the names `0` , `1` , and `3` represents the elements of the `greetings` object.
- The `length` property is initialized with a value of 3, indicating the number of elements in the `greetings` object.
- The `prepend()` method invokes the `call()` method of the `unshift()` method and sets the `this` to the `greetings` object. In other words, the `greetings` object borrows the `unshift()` method from an array object (`[]`).

Second, call the `prepend()` method of the `greetings` object to add an element at the index `0`th :

```
greetings.prepend('Good day');
```

Third, output the `greetings` object to the console:

```
console.log(greetings);
```

If you want to allow the `prepend()` method to add one or more elements to the `greetings` object, you can use the rest parameter and spread operator like this:

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

greetings.prepend('Good day', 'Bye');
```

In this example, the `prepend()` method accepts one or more messages (`...messages`) and passes them into the `unshift()` method individually using the spread operator.

Summary

- Use the JavaScript array `unshift()` method to add one or more elements to the beginning of an array.
- The `unshift()` method also works with the array-like object by using the `call()` or `apply()` method.