Array.prototype.push()

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

Introduction to the JavaScript Array push() method

The `Array.prototype.push()` method adds one or more elements at the end of an array and returns the new array's length.

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

```
push(newElement);
push(newElement1,newElement2);
push(newElement1,newElement2,...,newElementN);
```

The `push()` method modifies the original array and returns the new `length` property.

If you don't want to append one or more elements to an array but do not want to modify it, you can use the concat() method instead:

```
concat([newElement1, newElement2, ...]);
```

JavaScript Array push() method examples

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

1) Append one element to an array

The following example adds the number `40` to the end of the `numbers` array:

```
let numbers = [10, 20, 30];

const length = numbers.push(40);

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

Output:

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

How it works.

First, define the `numbers` array that has three numbers:

```
let numbers = [10, 20, 30];
```

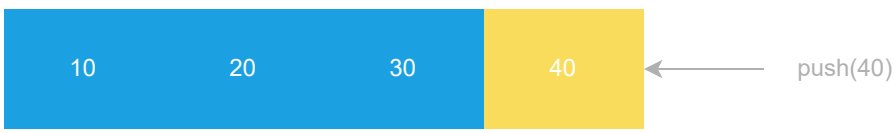
Second, add the number `40` to the end of the `numbers` array using the `push()` method and assign the return value to the `length` variable:

```
const length = numbers.push(40);
```

Third, output the `length` variable and the `numbers` array:

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

The following picture illustrates how the example works:



2) Add multiple elements to the end of an array

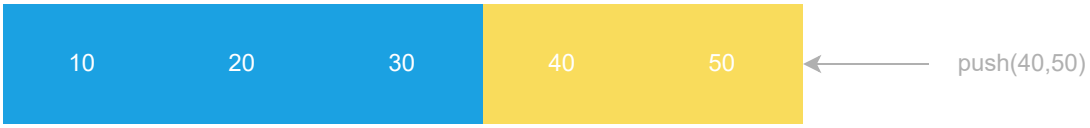
The following example uses the `push()` method to add multiple elements to the end of an array:

```
let numbers = [10, 20, 30];

const length = numbers.push(40, 50);

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

Output:

```
{ numbers: [ 10, 20, 30, 40, 50 ] }
{ length: 5 }
```

The following picture illustrates how it works:



3) Append elements of an array to another array

Suppose you have two arrays `colors` and `cmyk`:

```
let colors = ['red', 'green', 'blue'];
let cmyk = ['cyan', 'magenta', 'yellow', 'back'];
```

And you want to append the elements of the `cmyk` to the `colors` array.

To do that, you may use a `for...of` loop that iterates over the elements of the `cmyk` array and use the `push()` method to append each element to the `colors` array like this:

```
let colors = ['red', 'green', 'blue'];
let cmyk = ['cyan', 'magenta', 'yellow', 'back'];

for (const color of cmyk) {
  colors.push(color);
```

```
}

console.log(colors);
```

Output:

```
['red', 'green', 'blue', 'cyan', 'magenta', 'yellow', 'back']
```

Starting from ES6, you can use the spread operator (`...`) to spread the elements of the `cmyk` array and push them to the `colors` array at the same time like this:

```
let colors = ['red', 'green', 'blue'];
let cmyk = ['cyan', 'magenta', 'yellow', 'back'];

colors.push(...cmyk);

console.log({ colors });
```

Using the push() method with array-like objects

The `Array.prototype.push()` method is generic, meaning that you can call the `push()` method with the `call()` or `apply()` on array-like objects.

Under the hood, the `push()` method uses the `length` property to determine the position for inserting the elements.

If the `push()` method cannot convert the `length` property into a number, it'll use `0` as the value for the index.

See the following example:

```
let greetings = {
  0: 'Hi',
  1: 'Hello',
  length: 2,
  append(message) {
```

```
    [].push.call(this, message);
  },
};
greetings.append('Howdy');
greetings.append('Bonjour');

console.log(greetings);
```

Output:

```
{
  greetings: {
    '0': 'Hi',
    '1': 'Hello',
    '2': 'Howdy',
    '3': 'Bonjour',
    length: 4,
    append: [Function: append]
  }
}
```

How it works.

First, define the `greetings` object that has three properties `1`, `2`, and `length` and one method `append()`:

```
let greetings = {
  0: 'Hi',
  1: 'Hello',
  length: 2,
  append(message) {
    [].push.call(this, message);
  },
};
```

The `append()` method calls the `push()` method of an array object to append the `message` to the `greetings` object.

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

```
greetings.append('Howdy');
greetings.append('Bonjour');
```

In each call, the `push()` uses the `length` property of the `greetings` object to determine the position where it appends the new element and increases the `length` property by one.

As a result, the `greetings` object has two more elements at the index `2` and `3`. And the `length` property is `4` after the calls.

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

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

To allow the append() method to accept a number of messages, you can modify `append()` method like this:

```
let greetings = {
  0: 'Hi',
  1: 'Hello',
  length: 2,
  append() {
    [].push.call(this, ...arguments);
  },
};
greetings.append('Howdy', 'Bonjour');

console.log(greetings);
```

Output:

```
{
  '0': 'Hi',
  '1': 'Hello',
  '2': 'Howdy',
```

```
  '3': 'Bonjour',
  length: 4,
  append: [Function: append]
}
```

How it works.

First, remove the `message` parameter from the `append` method.

Second, spread out the elements of the `arguments` object and push them to the `greetings` object.

Summary

- Use the JavaScript array `push()` method to append one or more elements to an array.
- The `push()` method also works with an array-like object.