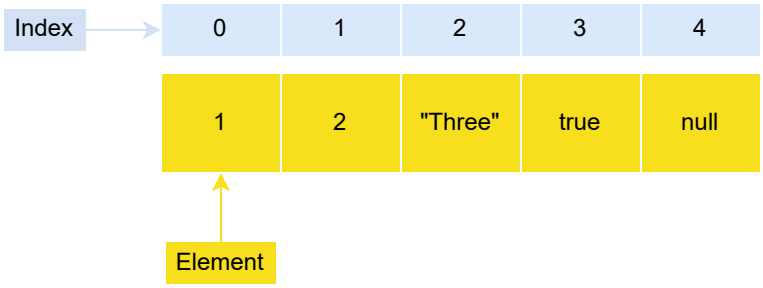
JavaScript Arrays

Summary: in this tutorial, you'll learn about JavaScript arrays and their basic operations.

Introduction to JavaScript arrays

In JavaScript, an array is an ordered list of values. Each value is called an *element* specified by an *index*:



A JavaScript array has the following characteristics:

1. First, an array can hold values of mixed types. For example, you can have an array that stores elements with the types number, string, boolean, and null.
2. Second, the size of an array is *dynamic* and *auto-growing*. In other words, you don't need to specify the array size up front.

Creating JavaScript arrays

JavaScript provides you with two ways to create an array. The first one is to use the `Array` constructor as follows:

```
let scores = new Array();
```

The `scores` array is empty, which does hold any elements.

If you know the number of elements that the array will hold, you can create an array with an initial size as shown in the following example:

```
let scores = Array(10);
```

To create an array and initialize it with some elements, you pass the elements as a comma-separated list into the `Array()` constructor.

For example, the following creates the `scores` array that has five elements (or numbers):

```
let scores = new Array(9, 10, 8, 7, 6);
```

> Note that if you use the `Array()` constructor to create an array and pass a number into it, you are creating an array with an initial size.

However, when you pass a value of another type like `string` into the `Array()` constructor, you create an array with an element of that value. For example:

```
let athletes = new Array(3); // creates an array with initial size 3
let scores = new Array(1, 2, 3); // create an array with three numbers 1,2 3
let signs = new Array('Red'); // creates an array with one element 'Red'
```

JavaScript allows you to omit the `new` operator when using the `Array()` constructor. For example, the following statement creates the `artists` array.

```
let artists = Array();
```

In practice, you'll rarely use the `Array()` constructor to create an array.

The more preferred way to create an array is to use the array literal notation:

```
let arrayName = [element1, element2, element3, ...];
```

The array literal form uses the square brackets `[]` to wrap a comma-separated list of elements.

The following example creates the `colors` array that holds string elements:

```
let colors = ['red', 'green', 'blue'];
```

To create an empty array, you use square brackets without specifying any element like this:

```
let emptyArray = [];
```

Accessing JavaScript array elements

JavaScript arrays are zero-based indexed. In other words, the first element of an array starts at index `0`, the second element starts at index `1`, and so on.

To access an element in an array, you specify an index in the square brackets `[]`:

```
arrayName[index]
```

The following shows how to access the elements of the `mountains` array:

```
let mountains = ['Everest', 'Fuji', 'Nanga Parbat'];

console.log(mountains[0]); // 'Everest'
console.log(mountains[1]); // 'Fuji'
console.log(mountains[2]); // 'Nanga Parbat'
```

To change the value of an element, you assign that value to the element like this:

```
let mountains = ['Everest', 'Fuji', 'Nanga Parbat'];
mountains[2] = 'K2';

console.log(mountains);
```

Output:

```
[ 'Everest', 'Fuji', 'K2' ]
```

Getting the array size

Typically, the `length` property of an array returns the number of elements. The following example shows how to use the `length` property:

```
let mountains = ['Everest', 'Fuji', 'Nanga Parbat'];
console.log(mountains.length); // 3
```

Basic operations on arrays

The following explains some basic operations on arrays. You'll learn advanced operations such as `map()`, `filter()`, and `reduce()` in the next tutorials.

1) Appending an element to an array

To add an element to the end of an array, you use the `push()` method:

```
let seas = ['Black Sea', 'Caribbean Sea', 'North Sea', 'Baltic Sea'];
seas.push('Red Sea');

console.log(seas);
```

Output:

```
[ 'Black Sea', 'Caribbean Sea', 'North Sea', 'Baltic Sea', 'Red Sea' ]
```

2) Adding an element to the beginning of an array

To add an element to the beginning of an array, you use the `unshift()` method:

```
let seas = ['Black Sea', 'Caribbean Sea', 'North Sea', 'Baltic Sea'];
seas.unshift('Red Sea');

console.log(seas);
```

Output:

```
[ 'Red Sea', 'Black Sea', 'Caribbean Sea', 'North Sea', 'Baltic Sea' ]
```

3) Removing an element from the end of an array

To remove an element from the end of an array, you use the `pop()` method:

```
let seas = ['Black Sea', 'Caribbean Sea', 'North Sea', 'Baltic Sea'];
const lastElement = seas.pop();
console.log(lastElement);
```

Output:

```
Baltic Sea
```

4) Removing an element from the beginning of an array

To remove an element from the beginning of an array, you use the `shift()` method:

```
let seas = ['Black Sea', 'Caribbean Sea', 'North Sea', 'Baltic Sea'];
const firstElement = seas.shift();

console.log(firstElement);
```

Output:

```
Black Sea
```

5) Finding an index of an element in the array

To find the index of an element, you use the `indexOf()` method:

```
let seas = ['Black Sea', 'Caribbean Sea', 'North Sea', 'Baltic Sea'];
let index = seas.indexOf('North Sea');
```

```
console.log(index); // 2
```

6) Check if a value is an array

To check if a value is an array, you use `Array.isArray()` method:

```
console.log(Array.isArray(seas)); // true
```

Quiz

Summary

- In JavaScript, an array is an order list of values; each value is called an element specified by an index.
- An array can hold values of mixed types.
- JavaScript arrays are dynamic, which means that they grow or shrink as needed.