Working With the Array Type

In this lesson, we'll cover the array type in JavaScript.
Let's begin!

WE'LL COVER THE FOLLOWING

- Creating arrays
- Quiz time! :)

If JavaScript has a jolly joker type beside Object, that is the Array type, without a doubt.

Just like in most programming languages, Array holds an **ordered list of data**, or a collection of items that can be indexed with an integer.

However,

JavaScript arrays are much more flexible than arrays in other languages.

Here's how:

> 1. An Array instance can hold **any type of data** in its items; the same array can hold numbers, objects, strings, or even arrays in its elements at the same time.
> 2. JavaScript arrays are **dynamically sized**, and automatically grow as data is added to them.

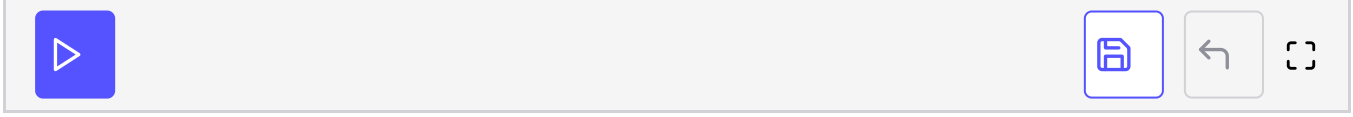
Creating arrays

You can easily create `new Array` instances with its constructor function, as this code snippet shows:

index.js

```
var emptyArr = new Array();
var arr6 = new Array(6);
var numbers = new Array("1", "two", 3);

console.log(emptyArr);
console.log(emptyArr.length);
console.log(arr6);
console.log(arr6.length);
console.log(numbers);
console.log(numbers.length);
```



This sample demonstrates that you can use the constructor function in **three different ways**.

➪ With passing no parameters, an empty array with zero-length is created.

➪ Using the constructor function with a number as the single argument creates an empty array and sets its length to the specified value.

➪ The third form of construction, where you pass more than one argument, creates an array initialized with the given elements. The output of the script indicates this fact:

console

```
[]
0
[]
6
["1", "two", 3]
3
```

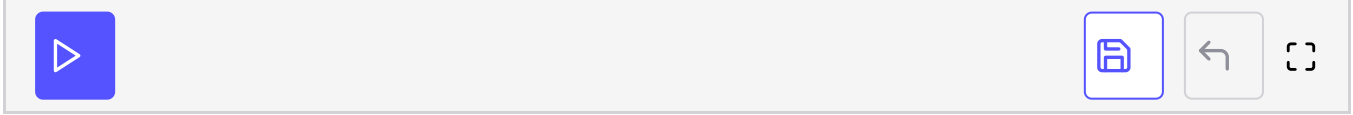
It shows that the first two arrays are empty; however, the second array's length is six.

Arrays are such a fundamental type in JavaScript that the language provides literals to construct a new Array instance. The next sample shows how to construct the same arrays as earlier, but this time using the array literal notation:

index.js

```
var emptyArr = [];
var arr6 = [,,,,,,];
var numbers = ["1", "two", 3];

console.log(emptyArr);
console.log(emptyArr.length);
console.log(arr6);
console.log(arr6.length);
console.log(numbers);
console.log(numbers.length);
```



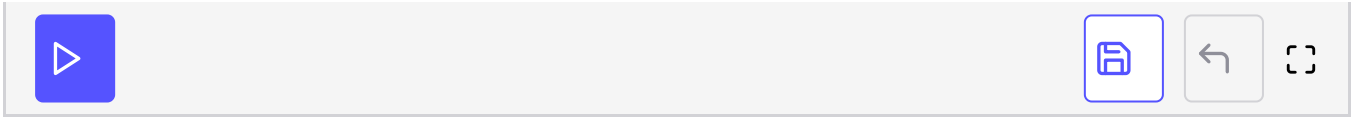
This code snippet produces the same output as the previous one. If you take a closer look at the `arr6` initialization you can count six commas, which means

there are seven elements. However, if the last comma is followed directly by a

closing square bracket, JavaScript ignores the last comma. You can try to change the declaration of numbers by adding a comma after 3:

index.js

```
var numbers = ["1", "two", 3,];
console.log(numbers);
```



This results in an array with three items, and this is why `arr6` represents an array with six items.

> 📜**NOTE:** Due to a bug 🐛, **Internet Explorer 8** does not handle the last comma in the `arr6` properly. It would create an array with seven items for `arr6`.

Quiz time! :)

It's time to test how much we've learned in this lesson with a short quiz!

Question 1

Q

```
var students = ["Velma", "Shaggy", "Scooby", "Fred"];
```

How would you access the second last element in the following `students` array?

Check Answers

Question 2

Q

The following statement:

```
student[0];
```

will access the *first* element in the array.

Check Answers

Great, now that we have covered the array type in detail, in the *next lesson*, let's delve into the art of array conversions in JavaScript.

See you there! :)