JavaScript Array Const

❮ Previous Next ❯

ECMAScript 2015 (ES6)

In 2015, JavaScript introduced an important new keyword: `const` .

It has become a common practice to declare arrays using `const` :

Example

```
const cars = ["Saab", "Volvo", "BMW"];
```

Try it Yourself »

Cannot be Reassigned

An array declared with `const` cannot be reassigned:

Example

```
const cars = ["Saab", "Volvo", "BMW"];
cars = ["Toyota", "Volvo", "Audi"];    // ERROR
```

Try it Yourself »

Arrays are Not Constants

The keyword `const` is a little misleading.

It does NOT define a constant array. It defines a constant reference to an array.

Because of this, we can still change the elements of a constant array.

Elements Can be Reassigned

You can change the elements of a constant array:

Example

```
// You can create a constant array:
const cars = ["Saab", "Volvo", "BMW"];

// You can change an element:
cars[0] = "Toyota";

// You can add an element:
cars.push("Audi");
```

Try it Yourself »

Browser Support

The `const` keyword is not supported in Internet Explorer 10 or earlier.

The following table defines the first browser versions with full support for the `const` keyword:

Chrome 49	IE 11 / Edge	Firefox 36	Safari 10	Opera 36
Mar, 2016	Oct, 2013	Feb, 2015	Sep, 2016	Mar, 2016

Assigned when Declared

JavaScript `const` variables must be assigned a value when they are declared:

Meaning: An array declared with `const` must be initialized when it is declared.

Using `const` without initializing the array is a syntax error:

Example

This will not work:

```
const cars;
cars = ["Saab", "Volvo", "BMW"];
```

Arrays declared with `var` can be initialized at any time.

You can even use the array before it is declared:

Example

This is OK:

```
cars = ["Saab", "Volvo", "BMW"];
var cars;
```

Try it Yourself »

Const Block Scope

An array declared with `const` has **Block Scope**.

An array declared in a block is not the same as an array declared outside the block:

Example

```
const cars = ["Saab", "Volvo", "BMW"];
// Here cars[0] is "Saab"
{
  const cars = ["Toyota", "Volvo", "BMW"];
  // Here cars[0] is "Toyota"
}
// Here cars[0] is "Saab"
```

Try it Yourself »

An array declared with `var` does not have block scope:

Example

```
var cars = ["Saab", "Volvo", "BMW"];
// Here cars[0] is "Saab"
{
  var cars = ["Toyota", "Volvo", "BMW"];
  // Here cars[0] is "Toyota"
}
// Here cars[0] is "Toyota"
```

Try it Yourself »

You can learn more about Block Scope in the chapter: JavaScript Scope.

Redeclaring Arrays

Redeclaring an array declared with `var` is allowed anywhere in a program:

Example

```
var cars = ["Volvo", "BMW"];   // Allowed
var cars = ["Toyota", "BMW"];  // Allowed
cars = ["Volvo", "Saab"];      // Allowed
```

Redeclaring or reassigning an array to `const`, in the same scope, or in the same block, is not allowed:

Example

```
var cars = ["Volvo", "BMW"];     // Allowed
const cars = ["Volvo", "BMW"];   // Not allowed
{
  var cars = ["Volvo", "BMW"];   // Allowed
  const cars = ["Volvo", "BMW"]; // Not allowed
}
```

Redeclaring or reassigning an existing `const` array, in the same scope, or in the same block, is not allowed:

Example

```
const cars = ["Volvo", "BMW"];   // Allowed
const cars = ["Volvo", "BMW"];   // Not allowed
var cars = ["Volvo", "BMW"];     // Not allowed
cars = ["Volvo", "BMW"];         // Not allowed

{
  const cars = ["Volvo", "BMW"]; // Allowed
  const cars = ["Volvo", "BMW"]; // Not allowed
  var cars = ["Volvo", "BMW"];   // Not allowed
  cars = ["Volvo", "BMW"];       // Not allowed
}
```

Redeclaring an array with `const`, in another scope, or in another block, is allowed:

Example

```
const cars = ["Volvo", "BMW"];   // Allowed
{
  const cars = ["Volvo", "BMW"]; // Allowed
}
{
  const cars = ["Volvo", "BMW"]; // Allowed
}
```

Complete Array Reference

For a complete Array reference, go to our:

Complete JavaScript Array Reference.

The reference contains descriptions and examples of all Array properties and methods.

❮ Previous

Next ❯