# JavaScript Variables

There are 3 ways to declare a JavaScript variable:

* Using var
* Using let
* Using const

## Variables

Variables are containers for storing data (values).

In this example, x, y, and z, are variables, declared with the var keyword:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Variables</h2>

<p>In this example, x, y, and z are variables.</p>

<p id="demo"></p>

<script>

var x = 5;

var y = 6;

var z = x + y;

document.getElementById("demo").innerHTML =

"The value of z is: " + z;

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Variables</h2>

<p id="demo"></p>

<script>

var price1 = 5;

var price2 = 6;

var total = price1 + price2;

document.getElementById("demo").innerHTML =

"The total is: " + total;

</script>

</body>

</html>

## JavaScript Identifiers

All JavaScript **variables** must be **identified** with **unique names**.

These unique names are called **identifiers**.

Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

The general rules for constructing names for variables (unique identifiers) are:

* Names can contain letters, digits, underscores, and dollar signs.
* Names must begin with a letter
* Names can also begin with $ and \_
* Names are case sensitive (y and Y are different variables)
* Reserved words (like JavaScript keywords) cannot be used as names

## The Assignment Operator

* In JavaScript, the equal sign (=) is an "assignment" operator, not an "equal to" operator.
* x = x + 5
* The "equal to" operator is written like == in JavaScript.

## JavaScript Data Types

JavaScript variables can hold numbers like 100 and text values like "John Doe".

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Variables</h2>

<p>Strings are written with quotes.</p>

<p>Numbers are written without quotes.</p>

<p id="demo"></p>

<script>

var pi = 3.14;

var person = "John Doe";

var answer = 'Yes I am!';

document.getElementById("demo").innerHTML =

pi + "<br>" + person + "<br>" + answer;

</script>

</body>

</html>

## Declaring (Creating) JavaScript Variables

Creating a variable in JavaScript is called "declaring" a variable.

You declare a JavaScript variable with the var keyword:

var carName;

After the declaration, the variable has no value (technically it has the value of undefined).

To **assign** a value to the variable, use the equal sign:

carName = "Volvo";

You can also assign a value to the variable when you declare it:

var carName = "Volvo";

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Variables</h2>

<p>Create a variable, assign a value to it, and display it:</p>

<p id="demo"></p>

<script>

var carName = "Volvo";

document.getElementById("demo").innerHTML = carName;

</script>

</body>

</html>

## One Statement, Many Variables

You can declare many variables in one statement.

Start the statement with var and separate the variables by **comma**:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Variables</h2>

<p>You can declare many variables in one statement.</p>

<p id="demo"></p>

<script>

var person = "John Doe", carName = "Volvo", price = 200;

document.getElementById("demo").innerHTML = carName;

</script>

</body>

</html>

A declaration can span multiple lines:

var person = "John Doe",  
carName = "Volvo",  
price = 200;

## Value = undefined

In computer programs, variables are often declared without a value. The value can be something that has to be calculated, or something that will be provided later, like user input.

A variable declared without a value will have the value undefined.

The variable carName will have the value undefined after the execution of this statement:

## Re-Declaring JavaScript Variables

If you re-declare a JavaScript variable, it will not lose its value.

The variable carName will still have the value "Volvo" after the execution of these statements:

### Example

var carName = "Volvo";  
var carName;

## JavaScript Arithmetic

As with algebra, you can do arithmetic with JavaScript variables, using operators like = and +:

JavaScript Let

The let keyword was introduced in [ES6 (2015)](https://www.w3schools.com/js/js_es6.asp).

Variables defined with let cannot be Redeclared.

Variables defined with let must be Declared before use.

Variables defined with let have Block Scope.

## Cannot be Redeclared

Variables defined with let cannot be **redeclared**.

You cannot accidentally redeclare a variable.

With let you can not do this:

let x = "John Doe";  
  
let x = 0;  
  
// SyntaxError: 'x' has already been declared

With var you can:

var x = "John Doe";  
  
var x = 0;

## Block Scope

Before ES6 (2015), JavaScript had only **Global Scope** and **Function Scope**.

ES6 introduced two important new JavaScript keywords: let and const.

These two keywords provide **Block Scope** in JavaScript.

Variables declared inside a { } block cannot be accessed from outside the block:

{  
  let x = 2;  
}  
// x can NOT be used here

Variables declared with the var keyword can NOT have block scope.

Variables declared inside a { } block can be accessed from outside the block.

{  
  var x = 2;  
}  
// x CAN be used here

## Redeclaring Variables

Redeclaring a variable using the var keyword can impose problems.

Redeclaring a variable inside a block will also redeclare the variable outside the block:

<!DOCTYPE html>

<html>

<body>

<h2>Redeclaring a Variable Using var</h2>

<p id="demo"></p>

<script>

var x = 10;

// Here x is 10

{

var x = 2;

// Here x is 2

}

// Here x is 2

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

Redeclaring a variable using the let keyword can solve this problem.

Redeclaring a variable inside a block will not redeclare the variable outside the block:

<!DOCTYPE html>

<html>

<body>

<h2>Redeclaring a Variable Using let</h2>

<p id="demo"></p>

<script>

let x = 10;

// Here x is 10

{

let x = 2;

// Here x is 2

}

// Here x is 10

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

The let keyword is not fully supported in Internet Explorer 11 or earlier.

# JavaScript Const

The const keyword was introduced in [ES6 (2015)](https://www.w3schools.com/js/js_es6.asp).

Variables defined with const cannot be Redeclared.

Variables defined with const cannot be Reassigned.

Variables defined with const have Block Scope

## Cannot be Reassigned

A const variable cannot be reassigned:

const PI = 3.141592653589793;  
PI = 3.14;      // This will give an error  
PI = PI + 10;   // This will also give an error

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript const</h2>

<p id="demo"></p>

<script>

try {

const PI = 3.141592653589793;

PI = 3.14;

}

catch (err) {

document.getElementById("demo").innerHTML = err;

}

</script>

</body>

</html>

## Must be Assigned

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

### Correct

const PI = 3.14159265359;

### Incorrect

const PI;  
PI = 3.14159265359;

## When to use JavaScript const?

As a general rule, always declare a variables with const unless you know that the value will change.

Always use const when you declare:

* A new Array
* A new Object
* A new Function
* A new RegExp

The keyword const is a little misleading.

It does not define a constant value. It defines a constant reference to a value.

Because of this you can NOT:

* Reassign a constant value
* Reassign a constant array
* Reassign a constant object

But you CAN:

* Change a constant array
* Change a constant object

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript const</h2>

<p>Declaring a constant array does NOT make the elements unchangeble:</p>

<p id="demo"></p>

<script>

// Create an Array:

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

// Change an element:

cars[0] = "Toyota";

// Add an element:

cars.push("Audi");

// Display the Array:

document.getElementById("demo").innerHTML = cars;

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript const</h2>

<p>You can NOT reassign a constant array:</p>

<p id="demo"></p>

<script>

try {

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

cars = ["Toyota", "Volvo", "Audi"];

}

catch (err) {

document.getElementById("demo").innerHTML = err;

}

</script>

</body>

</html>

## Constant Objects

You can change the properties of a constant object:

// You can create a const object:  
const car = {type:"Fiat", model:"500", color:"white"};  
  
// You can change a property:  
car.color = "red";  
  
// You can add a property:  
car.owner = "Johnson";

But you can NOT reassign the object:

const car = {type:"Fiat", model:"500", color:"white"};  
  
car = {type:"Volvo", model:"EX60", color:"red"};    // ERROR