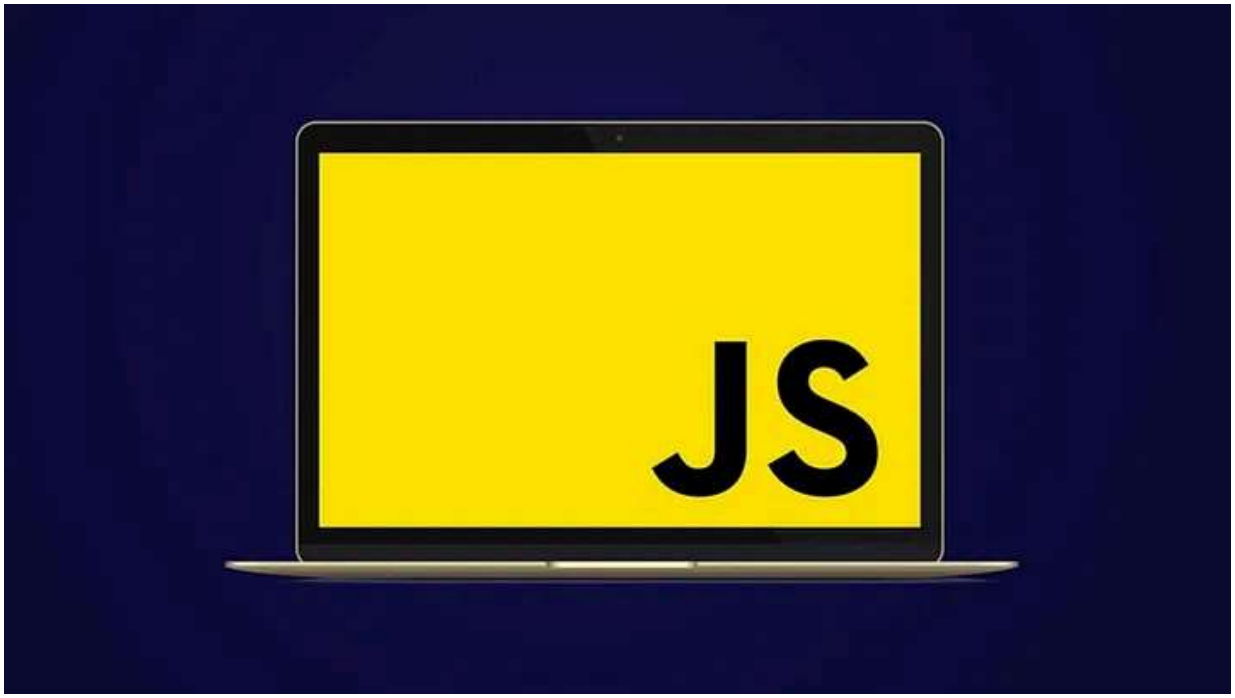
Reassignment / Redeclaration of Javascript Variables. Var, Let, Const.

James Robert Sutcliffe · Follow

4 min read · Jul 30, 2023



Variables in Javascript are containers use to store data types. **“var”, “let” and “const”** are the three main syntaxes used to declare variables in Javascript. Each syntax comes with different behaviour, scope and use-cases. In this article I will focus on each syntax’s behaviour in relation to it’s ability to mutate.

```
var x = "Hello";
let y = "World";
const z = 100;

// The above code lines are examples of declarations of variables.
// We use the variable syntax (var, let, const) followed by the name
// (x, y, z) then assign our value ("Hello", "World", 100) with the
// assignment operator "=".
```

These nuances can be tricky to get our head around at first, but once understood and applied correctly, our code will be adhering to modern coding-practices, readability and predictability.

Redeclaration / Reassignment of Variables

“var” is the traditional syntax used to declare variables, used from javascripts release in 1995 to 2015 (when let / const were introduced). **It is worth noting at this point that var should only be used in code written for legacy browsers, it is now considered good practice to exclude var and only apply “let” and “const”.**

“var” variables can be **redeclared** and **redefined**, meaning that once values are declared, they can be changed and mutated.

```
var x = 10;

console.log(x) // 10

// Reassign

var x = 20;

console.log(x) // 20

// Redeclare
```

```
x = 30;

console.log(x) // 30
```

The above snippet exemplifies how we can redeclare and reassign our declared variable. Web browsers read over javascript code by interpreting line by line, meaning as each variable is mutated as it is run through.

Although that may seem like a useful ability, it can be problematic. Think projects with thousands of lines of code and how easy it would be to ruin the flow of data with the accidental redeclaration of a named variable. Declaring with “let” is slightly different...

```
Reassign

let x = 10;

console.log(x)

let x = 20;

console.log(x)

// SyntaxError: Identifier 'x' has already been declared
```

When redeclaring a variable that has been declared with the “let” syntax an error message is received, javascript blocks this from happening. When our browser attempts to redeclare the let variable thecode stops running. However, Javascript does allow us to re-assign the value of variables...

```
// Reassign

let x= 10;

console.log(x, typeof x) // 10, number
```

```
x = "30";

console.log(x, typeof x) // 30, string
```

By assigning our already declared variable a new value, this becomes the new value of the variable. Once our browser has interpreted the line of code on which the reassignment occurs, the value will be reassigned. It is also worth noting that **the data type stored in variables is dynamic**, meaning we can reassign our variables to different data types.

Finally, declaring a variable with "const" declares the variable as **"read only"**...

```
const x = 10;

console.log(x)

let x = "30";

console.log(x)

//Identifier 'x' has already been declared

const x = 10;

console.log(x)

x = "30";

console.log(x)

// TypeError: Assignment to constant variable.
```

Similarly to "let", when we try to redeclare a "const" variable javascript blocks the code from running and logs an error message. Unlike "let", it also does the same when attempting to reassign a variable to a new value. It is not

the value itself that cannot be changed, but the reference to the value i.e. the name.

This is is made clear when destructuring objects / arrays, Javascript allows the reassignment of items / keys of objects / arrays declared with “const”.

```
const arr = ["a", "b", "c", "d"]

arr[1] = "F"

console.log(arr) // [ 'a', 'F', 'c', 'd' ]

 - - - - - - -

const room = {
    windows: 2,
    beds: 1,
    humans: 2
}

room.beds = 3
room.lamps = 2


console.log(room) // { windows: 2, beds: 3, humans: 2, lamps: 2 }
```

Variables that cannot be reassigned or redeclared are referred to as “Immutable” and those that can be are referred to as “mutable”. Think about their ability to mutate as opposed to speak!

Applying these “immutable” (const) and “mutuble” (let) variable syntaxes improve the readability of our code by allowing other programmers to identify whether our declared variables are intended to change. It is advisable to always apply “const” unless we know that our variable are going to be reassigned in which case we should apply “let”.

Choosing the right syntax for a particular situation can lead to more maintainable and bug-free code. However, it's important to understand the implications of each syntax and use them appropriately according to the requirements of your application.

To receive articles similar to this one to your inbox weekly please consider signing up to my FREE email newsletter.

The Coding Apprentice | James Robert Sutcliffe | Substack

Sign up for our FREE newsletter to receive a weekly article containing breakdowns of web development concepts and...

thecodingapprentice.substack.com

Follow me on twitter...

https://twitter.com/jrsCoding

JavaScript | Variables In Javascript | Coding | Programming | Learning To Code