Every value in JavaScript is one of the following types:

• A number

• The Boolean values false and true

• The special values null and undefined

• A string

• A symbol

• An object

The non-object types are collectively called *primitive types*.

**NOTE:** Even though the null type is distinct from the object type, typeof null is the string 'object'. This is a historical accident.

In JavaScript, variables do not have a type. You are free to store values of any type in any variable.

If you do not initialize a variable, it has the special value undefined:

If you never change the value of a variable, you should declare it with a const statement:

const PI = 3.141592653589793

If you try to modify the value contained in a const, a run-time error occurs.

You can declare multiple variables with a single const or let statement:

const FREEZING = 0, BOILING = 100

In strict mode, these keywords are also forbidden:

implements interface let package protected private public static

JavaScript has no explicit integer type. All numbers are double-precision floating-point numbers.

To convert a string to a number, you can use the parseFloat or parseInt functions

The toString method converts a number back to a string

It isn’t worth figuring out when a fractional number happens to work as an integer. If you are in such a situation, make your intent explicit by calling Math.trunc(x) to discard the fractional part, or Math.round(x) to round to the nearest integer.

Some number-producing functions return NaN to indicate a faulty input. For example, parseFloat('pie') is NaN.

Note that the / operator always yields a floating-point result, even if both operands are integers. For example, 1 / 2 is 0.5

The \*\* operator denotes “raising to a power,” as it does in Python

If s is a string and x a value of any type, then s + x and x + s are strings

To minimize confusion, it is a good idea to use actual Boolean values for all conditions.

JavaScript has two ways to indicate the absence of a value. When a variable is declared but not initialized, its value is undefined. This commonly happens with functions. When you call a function and fail to provide a parameter, the parameter variable has the value undefined.

The null value is intended to denote the intentional absence of a value.

The opposing point of view is that you should never set values to undefined and never return undefined from a function, but always use null for missing values. Then, undefined may signal a serious problem.

JavaScript uses the UTF-16 encoding which represents all Unicode code points with one or two 16-bit values called *code units*.

Template literals are strings that can contain expressions and span multiple lines. These strings are delimited by backticks (`. . .`).

You can nest template literals inside the ${. . .} expressions:

greeting = `Hello, ${firstname.length > 0 ? **`${firstname[0]}. `** : '' } ${lastname}`

A JavaScript object is simply a set of name/value pairs or “properties,” like this:

{ name: 'Harry Smith', age: 42 }

Such an object has only public data and neither encapsulation nor behavior. The object is not an instance of any particular class. In other words, it is nothing like an object in traditional object-oriented programming.

Use the delete operator to remove a property: delete harry.salary

Since arrays are objects, you can add arbitrary properties:

numbers.lucky = true

This is not common, but it is perfectly valid JavaScript.

The typeof operator returns 'object' for an array. To test whether an object is an array, call Array.isArray(obj).

**CAUTION:** JSON.stringify drops object properties whose value is undefined, and it turns array elements with undefined values to null. For example, JSON.stringify({ name: ['Harry', undefined, 'Smith'], age: undefined }) is the string '{"name":["Harry",null,"Smith"]}'.

**TIP:** To swap the values of the variables x and y, simply use: [x, y] = [y, x]

If you use destructuring for an assignment, the left-hand side doesn’t have to consist of variables. You can use any *lvalues*—expressions that can be on the left-hand side of an assignment. For example, this is valid destructuring:

[numbers[0], harry.age] = [13, 42] // Same as numbers[0] = 13; harry.age = 42