**There** are six basic **data types** in **JavaScript** which can be divided into three main categories: primitive (or primary), composite (or reference), and special **data types**. String, Number, and Boolean are primitive **data types**. Object, Array, and Function (which are all **types** of objects) are composite **data types**.

JavaScript variables can hold many **data types**: numbers, strings, objects and more:

var length = 16;                               // Number  
var lastName = "Johnson";                      // String  
var x = {firstName:"John", lastName:"Doe"};    // Object

JavaScript Booleans

Booleans can only have two values: true or false.

Example

var x = 5;  
var y = 5;  
var z = 6;  
(x == y)       // Returns true  
(x == z)       // Returns false

JavaScript Arrays

JavaScript arrays are written with square brackets.

Array items are separated by commas.

The following code declares (creates) an array called cars, containing three items (car names):

Example

var cars = ["Saab", "Volvo", "BMW"];

JavaScript Objects

JavaScript objects are written with curly braces {}.

Object properties are written as name:value pairs, separated by commas.

Example

var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

The typeof Operator

You can use the JavaScript typeof operator to find the type of a JavaScript variable.

The typeof operator returns the type of a variable or an expression:

Example

typeof ""             // Returns "string"  
typeof "John"         // Returns "string"  
typeof "John Doe"     // Returns "string"

Example

typeof 0              // Returns "number"  
typeof 314            // Returns "number"  
typeof 3.14           // Returns "number"  
typeof (3)            // Returns "number"  
typeof (3 + 4)        // Returns "number"

Undefined

In JavaScript, a variable without a value, has the value undefined. The type is also undefined.

Example

var car;    // Value is undefined, type is undefined

Any variable can be emptied, by setting the value to undefined. The type will also be undefined.

Example

car = undefined;    // Value is undefined, type is undefined

Empty Values

An empty value has nothing to do with undefined.

An empty string has both a legal value and a type.

Example

var car = "";    // The value is "", the typeof is "string"

Null

In JavaScript null is "nothing". It is supposed to be something that doesn't exist.

Unfortunately, in JavaScript, the data type of null is an object.

You can consider it a bug in JavaScript that typeof null is an object. It should be null.

You can empty an object by setting it to null:

Example

var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};  
person = null;    // Now value is null, but type is still an object

You can also empty an object by setting it to undefined:

Example

var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};  
person = undefined;   // Now both value and type is undefined

## Primitive Data

A primitive data value is a single simple data value with no additional properties and methods.

The typeof operator can return one of these primitive types:

* string
* number
* boolean
* undefined

### Example

typeof "John"              // Returns "string"  
typeof 3.14                // Returns "number"  
typeof true                // Returns "boolean"  
typeof false               // Returns "boolean"  
typeof x                   // Returns "undefined" (if x has no value)

## Complex Data

The typeof operator can return one of two complex types:

* function
* object

The typeof operator returns "object" for objects, arrays, and null.

The typeof operator does not return "object" for functions.

### Example

typeof {name:'John', age:34} // Returns "object"  
typeof [1,2,3,4]             // Returns "object”  
typeof null                  // Returns "object"  
typeof function myFunc(){}   // Returns "function"

The typeof operator returns "object" for arrays because in JavaScript arrays are objects.