JavaScript Data Types

❮ Previous Next ❯

JavaScript 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
```

The Concept of Data Types

In programming, data types is an important concept.

To be able to operate on variables, it is important to know something about the type.

Without data types, a computer cannot safely solve this:

```
var x = 16 + "Volvo";
```

Does it make any sense to add "Volvo" to sixteen? Will it produce an error or will it produce a result?

JavaScript will treat the example above as:

```
var x = "16" + "Volvo";
```

When adding a number and a string, JavaScript will treat the number as a string.

Example

```
var x = 16 + "Volvo";
```

Try it Yourself »

Example

```
var x = "Volvo" + 16;
```

Try it Yourself »

JavaScript evaluates expressions from left to right. Different sequences can produce different results:

JavaScript:

```
var x = 16 + 4 + "Volvo";
```

Result:

```
20Volvo
```

Try it Yourself »

JavaScript:

```
var x = "Volvo" + 16 + 4;
```

Result:

```
Volvo164
```

Try it Yourself »

In the first example, JavaScript treats 16 and 4 as numbers, until it reaches "Volvo".

In the second example, since the first operand is a string, all operands are treated as strings.

JavaScript Types are Dynamic

JavaScript has dynamic types. This means that the same variable can be used to hold different data types:

Example

```
var x;           // Now x is undefined
x = 5;           // Now x is a Number
x = "John";      // Now x is a String
```

Try it yourself »

JavaScript Strings

A string (or a text string) is a series of characters like "John Doe".

Strings are written with quotes. You can use single or double quotes:

Example

```
var carName = "Volvo XC60";   // Using double quotes
var carName = 'Volvo XC60';   // Using single quotes
```

Try it yourself »

You can use quotes inside a string, as long as they don't match the quotes surrounding the string:

Example

```
var answer = "It's alright";             // Single quote inside double
quotes
var answer = "He is called 'Johnny'";    // Single quotes inside double
quotes
var answer = 'He is called "Johnny"';    // Double quotes inside single
quotes
```

Try it yourself »

You will learn more about strings later in this tutorial.

JavaScript Numbers

JavaScript has only one type of numbers.

Numbers can be written with, or without decimals:

Example

```
var x1 = 34.00;     // Written with decimals
var x2 = 34;        // Written without decimals
```

Try it yourself »

Extra large or extra small numbers can be written with scientific (exponential) notation:

Example

```
var y = 123e5;      // 12300000
var z = 123e-5;     // 0.00123
```

Try it yourself »

You will learn more about numbers later in this tutorial.

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
```

Try it Yourself »

Booleans are often used in conditional testing.

You will learn more about conditional testing later in this tutorial.

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"];
```

Try it Yourself »

Array indexes are zero-based, which means the first item is [0], second is [1], and so on.

You will learn more about arrays later in this tutorial.

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"};
```

Try it Yourself »

The object (person) in the example above has 4 properties: firstName, lastName, age, and eyeColor.

You will learn more about objects later in this tutorial.

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"
```

Try it Yourself »

Example

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

Try it Yourself »

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
```

Try it Yourself »

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
```

Try it Yourself »

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"
```

Try it Yourself »

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
```

Try it Yourself »

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
```

Try it Yourself »

Difference Between Undefined and Null

Undefined and null are equal in value but different in type:

```
typeof undefined           // undefined
typeof null                // object

null === undefined         // false
null == undefined          // true
```

Try it Yourself »

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)
```

Try it Yourself »

Complex Data

The **typeof** operator can return one of two complex types:

- function
- object

The typeof operator returns object for both 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" (not "array", see note
below)
typeof null                  // Returns "object"
typeof function myFunc(){}   // Returns "function"
```

Try it Yourself »

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

❮ Previous Next ❯