

JavaScript Data Types

In this tutorial you will learn about the data types available in JavaScript.

Data Types in JavaScript

Data types basically specify what kind of data can be stored and manipulated within a program.

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. Whereas Undefined and Null are special data types.

Primitive data types can hold only one value at a time, whereas composite data types can hold collections of values and more complex entities. Let's discuss each one of them in detail.

The String Data Type

The *string* data type is used to represent textual data (i.e. sequences of characters). Strings are created using single or double quotes surrounding one or more characters, as shown below:

Example

[Try this code »](#)

```
var a = 'Hi there!'; // using single quotes
var b = "Hi there!"; // using double quotes
```

You can include quotes inside the string as long as they don't match the enclosing quotes.

Example

[Try this code »](#)

```
var a = "Let's have a cup of coffee."; // single quote inside double quotes
var b = 'He said "Hello" and left.'; // double quotes inside single quotes
var c = 'We\'ll never give up.'; // escaping single quote with backslash
```

You will learn more about the strings in [JavaScript strings](#) chapter.

The Number Data Type

The *number* data type is used to represent positive or negative numbers with or without decimal place, or numbers written using exponential notation e.g. $1.5e-4$ (equivalent to 1.5×10^{-4}).

Example

[Try this code »](#)

```
var a = 25;      // integer
var b = 80.5;    // floating-point number
var c = 4.25e+6; // exponential notation, same as 4.25e6 or 4250000
var d = 4.25e-6; // exponential notation, same as 0.00000425
```

The Number data type also includes some special values which are: Infinity, -Infinity and NaN. Infinity represents the mathematical Infinity ∞ , which is greater than any number. Infinity is the result of dividing a nonzero number by 0, as demonstrated below:

Example

[Try this code »](#)

```
alert(16 / 0); // Output: Infinity
alert(-16 / 0); // Output: -Infinity
alert(16 / -0); // Output: -Infinity
```

While NaN represents a special *Not-a-Number* value. It is a result of an invalid or an undefined mathematical operation, like taking the square root of -1 or dividing 0 by 0, etc.

Example

[Try this code »](#)

```
alert("Some text" / 2); // Output: NaN
alert("Some text" / 2 + 10); // Output: NaN
alert(Math.sqrt(-1)); // Output: NaN
```

You will learn more about the numbers in [JavaScript numbers](#) chapter.

The Boolean Data Type

The Boolean data type can hold only two values: `true` or `false`. It is typically used to store values like yes (`true`) or no (`false`), on (`true`) or off (`false`), etc. as demonstrated below:

Example

[Try this code »](#)

```
var isReading = true; // yes, I'm reading
var isSleeping = false; // no, I'm not sleeping
```

Boolean values also come as a result of comparisons in a program. The following example compares two variables and shows the result in an alert dialog box:

Example

[Try this code »](#)

```
var a = 2, b = 5, c = 10;

alert(b > a) // Output: true
alert(b > c) // Output: false
```

You will learn more about the comparisons in [JavaScript if/else](#) chapter.

The Undefined Data Type

The undefined data type can only have one value-the special value `undefined`. If a variable has been declared, but has not been assigned a value, has the value `undefined`.

Example

[Try this code »](#)

```
var a;
```

```
var b = "Hello World!"
```

```
alert(a) // Output: undefined
```

```
alert(b) // Output: Hello World!
```

The Null Data Type

This is another special data type that can have only one value-the `null` value. A `null` value means that there is no value. It is not equivalent to an empty string (`""`) or 0, it is simply nothing.

A variable can be explicitly emptied of its current contents by assigning it the `null` value.

Example

[Try this code »](#)

```
var a = null;
```

```
alert(a); // Output: null
```

```
var b = "Hello World!"
```

```
alert(b); // Output: Hello World!
```

```
b = null;
```

```
alert(b) // Output: null
```

The Object Data Type

The `object` is a complex data type that allows you to store collections of data. An object contains properties, defined as a key-value pair. A property key (name) is always a string, but the value can be any data type, like strings, numbers, booleans, or complex data types like arrays, function and other objects. You'll learn more about objects in upcoming chapters.

The following example will show you the simplest way to create an object in JavaScript.

Example

[Try this code »](#)

```
var emptyObject = {};  
var person = {"name": "Clark", "surname": "Kent", "age": "36"};  
  
// For better reading  
var car = {  
  "modal": "BMW X3",  
  "color": "white",  
  "doors": 5  
}
```

You can omit the quotes around property name if the name is a valid JavaScript name. That means quotes are required around `"first-name"` but are optional around `firstname`. So the car object in the above example can also be written as:

Example

[Try this code »](#)

```
var car = {  
  modal: "BMW X3",  
  color: "white",  
  doors: 5  
}
```

You will learn more about the objects in [JavaScript objects](#) chapter.

The Array Data Type

An array is a type of object used for storing multiple values in single variable. Each value (also called an element) in an array has a numeric position, known as its index, and it may contain data of any data type-numbers, strings, booleans, functions, objects, and even other arrays. The array index starts from 0, so that the first array element is `arr[0]` not `arr[1]`.

The simplest way to create an array is by specifying the array elements as a comma-separated list enclosed by square brackets, as shown in the example below:

Example

[Try this code »](#)

```
var colors = ["Red", "Yellow", "Green", "Orange"];
var cities = ["London", "Paris", "New York"];
```

```
alert(colors[0]); // Output: Red
alert(cities[2]); // Output: New York
```

You will learn more about the arrays in [JavaScript arrays](#) chapter.

The Function Data Type

The function is callable object that executes a block of code. Since functions are objects, so it is possible to assign them to variables, as shown in the example below:

Example

[Try this code »](#)

```
var greeting = function(){  
    return "Hello World!";  
}  
  
// Check the type of greeting variable  
alert(typeof greeting) // Output: function  
alert(greeting());    // Output: Hello World!
```

In fact, functions can be used at any place any other value can be used. Functions can be stored in variables, objects, and arrays. Functions can be passed as arguments to other functions, and functions can be returned from functions. Consider the following function:

Example

[Try this code »](#)

```
function createGreeting(name){  
    return "Hello, " + name;  
}  
  
function displayGreeting(greetingFunction, userName){  
    return greetingFunction(userName);  
}  
  
var result = displayGreeting(createGreeting, "Peter");  
alert(result); // Output: Hello, Peter
```

